DAS Departamento de Automação e Sistemas
CTC Centro Tecnológico
UFSC Universidade Federal de Santa Catarina

Projeto de um Clock Disciplinado Auto-Ajustável em Tempo Real

Relatório submetido à Universidade Federal de Santa Catarina

como requisito para a aprovação na disciplina

DAS 5511: Projeto de Fim de Curso

Brunno Abner Machado

Florianópolis, fevereiro de 2013

Projeto de um Clock Disciplinado Auto-Ajustável em Tempo Real

Brunno Abner Machado

Esta monografia foi julgada no contexto da disciplina
DAS5511: Projeto de Fim de Curso
e aprovada na sua forma final pelo
Curso de Engenharia de Controle e Automação

Prof. *Joni da Silva Fraga*

_____
Assinatura do Orientador

Banca Examinadora:

Celso Souza / Reason Tecnologia
Orientador na Empresa

Prof. Joni da Silva Fraga
Orientador no Curso

Agradecimentos

Primeiramente aos meus pais, Paulo Roberto Machado e Solange Regina Machado e a minha irmã Agnes Cristina Machado, por todos os conselhos e ensinamentos, e pelo apoio e incentivo durante toda a minha graduação ao longo de toda minha vida.

Agradeço a todos os professores do curso de Engenharia de Controle e Automação pelos conhecimentos repassados e pela dedicação curso.

Agradeço aos meus orientadores Joni da Silva Fraga e Celso Souza pelos conselhos e contribuições que muito ajudaram no desenvolvimento deste trabalho.

Para finalizar, agradeço toda equipe do departamento de desenvolvimento da Reason, por propiciarem um ambiente muito agradável e incentivador para realização desse trabalho, e por todo suporte oferecido. Em especial aos colegas Lucas Silveira Groposo e Marcelo Dalmas.

Resumo

Redes de transmissão de energia possuem sistemas de aquisição que precisam se manter sempre sincronizados. Estas sincronizações são realizadas com auxílio do GPS e garantem a consistência dos dados.

A empresa Reason Tecnologia S/A, já é consagrada nas áreas de desenvolvimento de equipamentos para oscilografia, qualidade de energia e sincronismo temporal. Atualmente está desenvolvendo produtos na linha de proteção elétrica para subestações e basicamente todas as linhas de produtos se beneficiarão com a melhoria da precisão fornecida por um relógio disciplinado.

Um relógio disciplinado é um circuito que opera em malha fechada, aonde a frequência de saída é controlada e com isso a precisão alcançada é muito maior. Além de mais preciso um clock disciplinado é mais robusto, pois compensa as variações intrínsecas aos cristais geradores de clock.

No presente trabalho foi realizado na empresa Reason e consistiu na idealização, projeto e implementação de um circuito lógico digital que controla a operação de um oscilador Si599, criando assim, um clock disciplinado. Esse circuito foi construído em um FPGA utilizando a linguagem de descrição de hardware VHDL.

Como resultado o projeto atingiu as expectativas, respeitando o erro máximo de 100ns por segundo e retornando a frequência desejada sem saltos com a volta do sinal de pps.

Abstract

Power transmission networks have acquisition systems that must always remain synchronized. These synchronizations are performed with the aid of GPS and ensure data consistency.

The company Reason Technology S / A, is already consecrated in the areas of development of equipment for oscillography, power quality and temporal synchronization. Is currently developing products in line of protection for electrical substations and basically all product lines will benefit from the improved accuracy provided by a disciplined clock.

A disciplined clock a closed loop circuit where the output frequency is controlled and thus the accuracy achieved is much higher. In addition to having greater accuracy, a disciplined clock is more robust because it compensates the crystal clock generators variations.

The present work was performed in Reason Technology and was consisted of idealization, design and implementation of a digital logic circuit that controls the operation of an oscillator Si599, thus creating a disciplined clock. This circuit was built on an FPGA using a hardware description language VHDL.

As a result the project met expectations, respecting the maximum error of 100ns per second and returning to the desired frequency without jumps on the return of pps signal.

Sumário

Indice de figuras

Simbologia

DCO	Digital Controlled Oscillator
FPGA	Field Programmable Gate Array
GPS	Global Position System
GPST	GPS Time
HDL	Hardware Description Language
IEC	International Electrotechnical Comission
IEEE	Institute of Electrical and Electronics Engenieers
IP	Intelectual Property
PPS	Pulso Por Segundo
PTP	Precision Time Protocol
RP	Registrador de Perturbação
RTL	Register Transfer Level
ONS	Operador Nacional do Sistema
SAS	Sistema de Automação de Subestação
SDI	Serial Digital Interface
SEB	Setor Elétrico Brasileiro
SIN	Sistema Interligado Nacional
TAI	Temps Atomique International
VCO	Voltage Controlled Oscillator

VHDL Very High Speed Integrated Circuit HDL

Capítulo 1: Introdução

A sociedade atual se desenvolveu tendo como base o uso de energia elétrica. Basicamente todos os equipamentos modernos utilizam energia elétrica de alguma forma, com isso surge à necessidade cada vez maior de geração e distribuição dessa eletricidade. Essa energia é gerada em usinas (hidroelétricas, termoelétricas, nucleares etc.) que convertem energia das mais variadas formas em energia elétrica.

Já o transporte da energia gerada em usinas até o consumo em industrial e residencial acontece por meio de linhas de transmissão de alta tensão, que podem se estender por centenas de quilômetros de uma ponta a outra. Para realizar essa interface entre as linhas de transmissão e os consumidores finais, assim como para identificar falhas de transmissão e realizar a proteção da rede elétrica existem instalações conhecidas como subestações.

Uma vez gerada e transmitida essa energia elétrica é distribuída e comercializada para chegar ao consumidor final.

Visto que a eletricidade é crucial para o funcionamento da sociedade moderna, é extremamente importante que as linhas de transmissão permaneçam funcionando de forma segura e com o mínimo de interrupções para que o consumidor continue a ter eletricidade disponível para o uso. Além disso, em caso de falhas ou interrupções é importante que se tenha a informação precisa de em sua análise. Estas para que essa precisão seja adequada é crucial que equipamentos em ambos os lados da transmissão estejam sincronizados entre si.

Como eventos elétricos ocorrem em escalas de tempo na ordem de milissegundos, esta sincronização precisa ser feita em nível de relógio. A melhor forma de sincronizar relógios separados por centenas de quilômetros é a sincronização externa com um relógio de referência real e extremamente preciso. Isto é feito sincronizando-se cada equipamento ao GPS que possui uma precisão da ordem de 14ns.

Alguns equipamentos são regidos por normas, uma dessas normas é a IEC61588 Ed2.0 onde é apresentado o padrão IEEE1588 ou PTP (Precision Time Protocol). Além do PTP existem outros padrões como o IRIG-B e o PPS (Pulso Por Secundo). Independente do padrão usado, um circuito gerador de clock deve ser capaz de seguir a referência e funcionar da melhor forma possível em caso de ausência de referência.

Um clock disciplinado é um circuito gerador de clock operando em malha fechada, que pode se auto-ajustar em tempo real para seguir uma referência. Isto faz com que as variações na geração do clock sejam compensadas e o clock de saída seja mais preciso e estável.

1.1: Motivação, Objetivos e Justificativas

A Reason Tecnologia S/A é uma empresa que é referência em equipamentos da área de oscilografia, qualidade de energia, sincronismo temporal e mais recentemente relés de proteção. Desenvolvendo relés de proteção, a Reason anseia em um novo mercado, o de proteção elétrica, e precisa oferecer produtos com um diferencial em relação aos presentes no mercado a fim de conquistar clientes e se firmar nesse novo ambiente. Observando as tendências do setor de sincronismo temporal e buscando sempre estar a frente da concorrência, a Reason reconheceu a importância de que seus novos produtos continuem inovando oferecendo novos padrões de precisão aprimorando o sincronismo.

1.2: Metodologia Utilizada

Esse trabalho foi realizado em várias etapas. São elas:

- Estudo dos padrões de desenvolvimento da empresa e adequação das estruturas de programação: Nessa etapa, fez-se uma leitura das literaturas de referência usadas na descrição de hardware em VHDL utilizadas na empresa.

- Levantamento dos requisitos funcionais e não funcionais: Nessa etapa listou-se o que o clock disciplinado deveria ser capaz de realizar e as restrições quanto ao seu funcionamento.
- Baseado nos requisitos, análise e formulação da solução proposta: Nessa etapa definiu-se que o clock disciplinado seria criado como circuito lógico digital e, baseado nessa escolha, foi definido quais ferramentas seriam utilizadas.
- Aplicação prática da solução proposta.
- Testes e avaliação do resultado.

1.3: Estrutura do documento

Segue abaixo a organização desse documento:

O capítulo 2 fará uma contextualização do ambiente em que esse trabalho se insere, descrevendo melhor as motivações e objetivos. Será apresentado também a empresa Reason.

O capítulo 3 serão abordados conceitos relacionados ao projeto. Nesse capítulo será apresentado o oscilador Si599, como se comunica, o protocolo I²C e conceitos sobre sincronismo.

O capítulo 4 apresentará as especificações quanto à realização do clock disciplinado e a escolha da tecnologia utilizada para implementação.

O capítulo 5 mostra toda a etapa do projeto do circuito lógico digital do clock disciplinado, apresentando diagramas que mostram quais componentes estão presentes e como eles se relacionam, e também máquinas de estados que descrevem o funcionamento de cada módulo.

O capítulo 6 fará uma descrição de como o projeto definido no capítulo 5 foi implementado. Este capítulo também irá apresentar os testes realizados e os resultados obtidos.

No capítulo 7 fará uma conclusão do trabalho e apresentação de perspectivas futuras.

Capítulo 2: Contextualização

2.1: Transmissão de energia

De acordo com as Leis nº 10.847 e 10.848, de 15 de março de 2004, e pelo Decreto nº 5.163, de 30 de julho de 2004, o governo federal atualizou o modelo para o Setor Elétrico Brasileiro (SEB) [1] .

O novo modelo do setor elétrico visa atingir três objetivos principais:

- Garantir a segurança do suprimento de energia elétrica;
- Promover a modicidade tarifária;
- Promover a inserção social no Setor Elétrico Brasileiro, em particular pelos programas de universalização de atendimento.

A geração de energia elétrica geralmente ocorre em locais distantes de onde esta energia será utilizada. Com isso grandes redes de transmissão de energia são necessárias para atender todo o Sistema Interligado Nacional (SIN).

Qualquer equipamento está sujeito a falhas, e com as linhas de transmissão isso não é diferente. Contudo, linhas de transmissão podem ter mais de 500 km de comprimento a localização dessas falhas é crucial para que sejam evitados longos períodos sem transmissão da energia.

2.2: Medições

Com a ocorrência de falhas e ruídos nas linhas de transmissão, ocorrem variações na frequência, tanto em seu módulo, quanto em sua fase. A captação dessas variações e o seu entendimento é fundamental para identificação, correção ou até mesmo prevenção de falhas.

A oscilografia pode ser descrita como o estudo da forma de onda de grandezas elétricas como corrente, tensão e potência, presentes na transmissão de energia elétrica. Para monitorar os regimes transitórios presentes em uma linha de transmissão e visualizar suas formas de onda existem equipamentos chamados de oscilógrafos.

Por volta dos anos 50 surgiram os primeiros oscilógrafos para identificar falhas em redes elétricas. Como a tecnologia da época era analógica, esses equipamentos geravam dados em papel. Esta tecnologia foi atualizada nos anos 70 para impressão em papel fotográfico e nos anos 80 e 90 para a tecnologia digital presente até hoje. Embora seja predominante a utilização de tecnologia digital nos dias de hoje, algumas subestações ainda apresentam equipamentos analógicos.

O estudo da oscilografia é complementar ao estudo da proteção de sistemas elétricos, uma vez que possibilita a correção dos ajustes de proteção, baseando-se nos registros obtidos com o oscilógrafo.

Outro conceito importante na aquisição de dados em medições realizadas em sistemas de transmissão de energia é o conceito de ondas viajantes (travelling waves). Esta metodologia é utilizada para se identificar falhas em linhas de transmissão e precisar a sua localização ao longo da linha. A falha se propaga ao longo da linha de transmissão levando tempos diferentes de propagação para cada lado da linha, assim é possível identificar o local da falha com precisão de 300 metros ao logo de linhas com centenas de quilômetros de comprimento.

Contudo, para que se tenha a precisão necessária é indispensável que os sistemas de aquisição estejam sincronizados. Mais sobre medições pode ser encontrado em [2].

2.3: Sincronização de dados

O método de sincronização utilizado pela Reason é eficiente e preciso. Baseia-se na sincronização através do Sistema de Posicionamento Global. Este por sua vez é uma referência de clock real com precisão da ordem de 14ns. Seu sistema

de relógios é o GPST (Global Position System Time) e é baseado no TAI (Temps Atomique International, em Francês). O Tempo Atômico internacional consiste em uma série de relógios atômicos espalhados pelo mundo que se comunicam e geram uma base de tempo única.

Embora esse sistema de referência temporal seja muito preciso, devido a uma série de fatores, o sinal de pps proveniente do GPS não é sempre recebido. Isso faz com que o relógio local comece a divergir da sua frequência de referência. Isto causa problemas de sincronismo e erros de medições.

2.4: Clocks disciplinados

Enquanto uma referência de tempo real esteja disponível, um circuito gerador de clock para sistemas de sincronismo deve atender a especificações e seguir padrões para garantir precisão desses sistemas. Um desses padrões é o IEE1588 [3] ou PTP, que estabelece que a taxa de deriva (detalhada na seção 3.1.4:) limite é de 100ns por segundo.

Além do funcionamento correto quando uma referência está disponível, um sistema de sincronia precisa se manter constante com a ausência da referência. Com a volta da referência, a frequência gerada pelo relógio de sincronismo não estará mais dentro dos padrões exigidos e deverá ser corrigida automaticamente, entretanto caso essa correção seja realizada de forma abrupta, o sistema de medição irá apresentar um erro. Este erro de medição se decorre do fato de um salto na frequência poder ser interpretado como um pico na fase, e assim algum equipamento, como um relé de proteção, pode interpretar isso como uma falha e abrir uma linha de transmissão sem que nenhum problema real estivesse ocorrendo.

Para evitar esse comportamento o circuito gerador de clock deve ser capaz de voltar a frequência de referência obedecendo a um limite de salto bem determinado.

2.5: A empresa Reason Tecnologia S/A

A empresa Reason Tecnologia S/A surgiu em 1991 com a sua fundação em uma incubadora de empresa em Florianópolis, Santa Catarina, Brasil, onde atualmente está a sua sede.

Sua área de atuação é principalmente o setor de transmissão de energia, onde desenvolve soluções para oscilografia, qualidade de energia e sincronismo temporal. Seus principais clientes são as companhias de geração, transmissão e distribuição de energia elétrica, grandes indústrias, empresas de engenharia, integradores e prestadores de serviço. Entre os seus clientes podemos citar: Celesc, Eletrosul, Copel, Chesf entre outras.

Seu primeiro produto foi o Registrador Digital de Perturbações RP-II. Em 1995 ingressou na Tecnópolis, onde lançou o Registrador Digital de Perturbações RP-III. Em 1997 lançou o RP-IV que se tornou o principal equipamento de oscilografia do Brasil, tornando-se referência nacional em tecnologia para o setor elétrico. Em 2007 lançou o RPV Registrador Digital Multifunção com a tecnologia inovadora da função de localização de faltas por ondas viajantes, que atualmente, é a tecnologia de melhor precisão e eficiência para reduzir o tempo de falta do sistema. Ainda em 2007 a empresa abriu uma filial nos Estados Unidos e, em paralelo, forneceu equipamentos para vários países da América Latina.

Em 2008 a Reason desenvolveu o RT420 Relógio Sincronizado por Satélites, que concentra todas as funcionalidades de outros receptores GPS comercializados anteriormente em um único equipamento. No final de 2008, a empresa abriu uma filial na Europa, em Berlim na Alemanha. Em 2009, lançou produtos voltados especialmente para monitoração e localização de faltas por ondas viajantes, e Registradores Digitais de Perturbações menores, mas mantendo as funcionalidades e características desenvolvidas para a família RPV. Em 2011 recebeu o prêmio FINEP de Inovação, competindo com empresas de todo o país. Atualmente além de se manter como referência mundial no segmento desenvolve um relé de proteção para atuar em um novo mercado. Mais em [4].

2.5.1: Linha de Produtos

A Reason oferece linhas de produtos na área de Oscilografia com Registradores Digitais de Perturbações (RDP) e na área de sincronismo temporal com Relógios Sincronizados por GPS [5].

O Registrador Digital de Perturbações executa funções de oscilografia, além de outras requeridas atualmente para análise de uma ou mais instalações elétricas. Além do tradicional registro de forma de onda utilizado para análise de curtos, energização e outros transitórios, outras funcionalidades como registros fasoriais, registros de qualidade de energia, medição e difusão de sincrofasores também estão presentes.

Os produtos da linha de sincronismo temporal disponibilizam uma base de tempo única, referida ao sistema GPS, que permitem a análise de informações de diversos equipamentos. A linha de sincronismo temporal compreende os equipamentos RT420 (Relógio Sincronizado por Satélites GPS), RT412 (Transceptor Óptico), RT1320 (Repetidor de Tempo), RT1400 (Display IRIG-B).

Recentemente a empresa está desenvolvendo produtos na linha de Relés de Proteção Elétrica, ingressando em um novo mercado.

2.5.2: Os Projetos relacionados

A primeira utilização do clock disciplinado será na área de sincronismo temporal com os Relógios Sincronizados por GPS, porém por se tratar de uma solução de baixo custo e ser idealizado para a utilização em sistemas sem processadores, pode ser incluído em praticamente todos os projetos da empresa.

Além disso, com a melhoria na precisão proporcionada, será possível desenvolver equipamentos de telecomunicações. Abrindo assim as portas de mais um setor para investimento da empresa.

2.6: Contextualização com o curso

O presente documento foi desenvolvido como atividade para o Projeto de Fim de Curso (DAS 5511), no departamento de desenvolvimento da empresa Reason Tecnologia S/A. Uma série de disciplinas do curso estão relacionadas com os trabalhos realizados no projeto em maior ou menor grau. Sendo que a mais importante no presente contexto é a disciplina de Sistemas Digitais. Outras disciplinas como: Microprocessadores, Informática Industrial I e II, Eletrônica e Sistemas Distribuídos, Sinais e Sistemas Lineares 1, Sinais e Sistemas Lineares 2, Sistemas Realimentados, Sistemas Não-Lineares e Instrumentação em Controle; também contribuíram com diversos conceitos para melhor entendimento do problema e elaboração de sua solução

2.7: Conclusão

Neste capítulo foi feito uma descrição do atual cenário e ambiente em que este trabalho se insere. Foi feito também uma apresentação da empresa onde o trabalho foi realizado, assim como a motivação para o mesmo.

Nós próximos capítulos serão apresentados o embasamento teórico necessário para o entendimento do projeto.

Capítulo 3: Conceitos relacionados

O capítulo 3 fornecerá, inicialmente, embasamento teórico necessário para a crompreensão do clock disciplinado e seu funcionamento.

3.1: Sincronismo

O sincronismo é a garantia de que processos diferentes concordem com a ordem em que eventos distintos ocorreram. Tanto em sistema de baixo nível quanto sistemas de alto nível utilizam técnicas de sincronização para que tenham consistência nos dados em sistemas multi-agentes.

3.1.1: Sistema síncrono

De acordo com [6] em um sistema síncrono existem limites de tempo bem definidos nas trocas de informação. Este tipo de sistema se caracteriza por possuir:

- Limite entre atraso relativo de processamento
- Limite máximo na latência de uma mensagem na comunicação

3.1.2: Sistema assíncrono

Já nos sistemas assíncronos, não existe um limite de tempo para a troca de informações e muito menos para as latências de comunicação.

3.1.3: Ordenação de mensagens por relógios físicos

Para se ordenar uma comunicação em dois ou mais pontos através de relógios físicos, é necessário que ambos tenham acesso a um sinal de clock global de tempo-real.

As mensagens trocadas desse modo recebem um *timestamp* que identifica o momento exato no tempo em que a mensagem ou evento ocorreu. Com base nos timestamps é possível realizar um ordenamento de acordo a informação temporal contida no timestamp e observando o erro δ máximo que representa a variação máxima entre os clock.

Esta necessidade de sincronização com um relógio externo elimina os problemas de sincronização entre relógios em sistemas distribuídos onde constantemente entram ou saem novos agentes comunicadores. Assim cada elemento na rede de comunicação precisa garantir que o seu clock interno está sincronizado com a referência, respeitando um limite de erro máximo definido.

3.1.4: Taxa de deriva

Por mais preciso que seja o clock de um sistema, ele nunca será igual ao clock de outro sistema. Isso se deve graças às limitações físicas da construção e funcionamento dos cristais geradores de clock. Um clock sincronizado com uma referência sempre se afastará desta referência ao passar do tempo. Este afastamento do clock ideal de referencia é conhecido como *taxa de deriva ρ (drift rate).* Para que um clock interno a um sistema seja considerado sincronizado com sua referência, a taxa de deriva deve ser inferior a um valor determinado.

A Figura 1 ilustra esse comportamento, onde Ci(t) é o tempo representado pelo relógio interno.

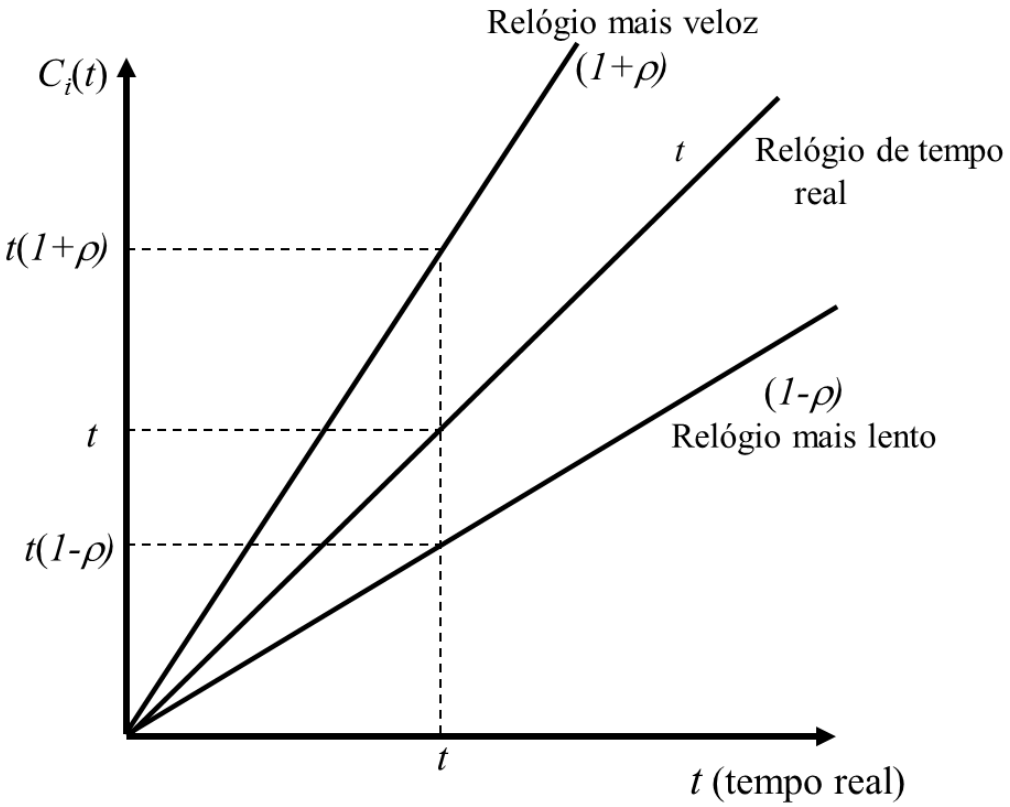


Figura 1 Taxa de deriva(Drift Rate)

3.1.5: Melhoria de um clock disciplinado

Uma vez conhecido o conceito de taxa de deriva, fica simples entender as melhorias que um clock disciplinado proporciona. A Figura 2 mostra o comportamento temporal de um clock disciplinado. É possível notar que, ao contrario do que acontece em clocks não controlados automaticamente, o clock disciplinado sempre se mantem dentro de uma faixa de erro à referência invariante no tempo. Este comportamento é observado enquanto a referencia de sincronia estiver disponível e é uma consequência de um clock disciplinado operar em malha fechada.

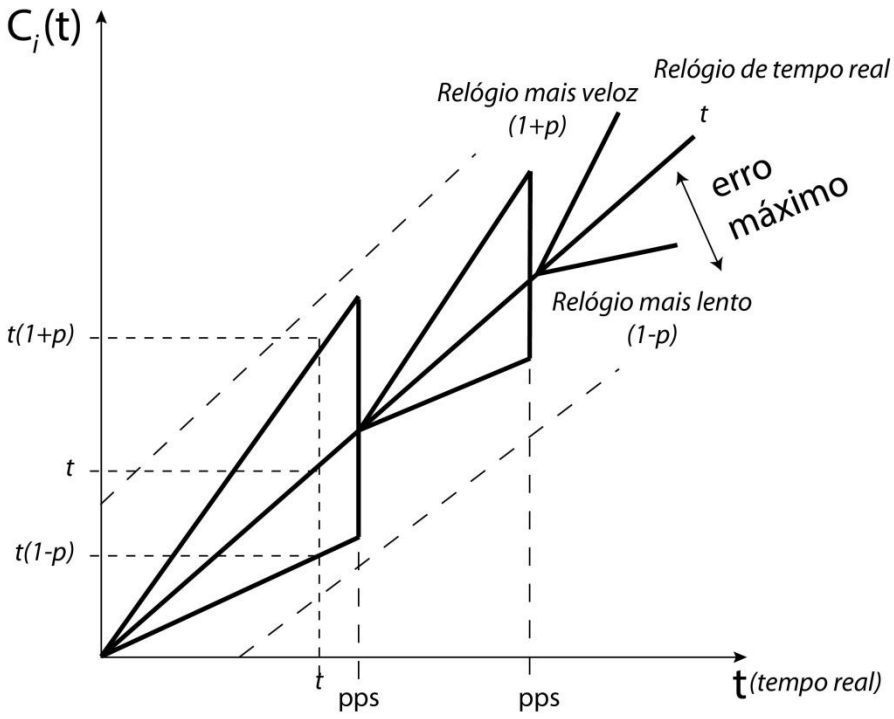


Figura 2 Deriva no Clock Disciplinado

3.1.6: Referência de tempo real

Existem basicamente dois principais métodos para se medir o tempo real. O método mais antigo se baseia no movimento da Terra (tempo astronômico). O grande problema desse método é que os segundos variam de acordo com a posição da Terra.

Atualmente os sistemas de precisão se baseiam em um novo método de medir o tempo. Este método utiliza osciladores atômicos ao invés de o posicionamento dos astros. O relógio físico mais famoso que utiliza esse método de medição é o TAI.

3.2: Oscilador Si599

O Si599 é um oscilador com baixo jitter idealmente projetado para aplicações que necessitam de uma frequência programável. Ele pode ser programado para gerar qualquer clock de saída entre o intervalo de 10 à 810MHz com uma resolução de frequência de 30 partes por trilhão. A performance do jitter de saída é suficiente para atender as excede os requerimentos exigidos de performance de comunicações de alta velocidade incluindo OC-48/STM-16, 3G SDI, e Gigabit Ethernet.

O Si599 consiste em um oscilador controlado digitalmente (DCO) baseado na terceira geração da tecnologia DSPLL do Silicon Laboratories [7], que é formado por um cristal de relógio interno com frequência fixa como referência.

A frequência de saída padrão do dispositivo é ajustada de fábrica e pode ser reprogramada através de uma porta serial I²C de dois fios. Uma vez que o oscilador seja desligado ele volta a sua frequência padrão de fábrica.

3.3: Protocolo de comunicação I²C

O I²C (“I” ao quadrado “c” ou “i” dois “c”) é um barramento serial de dois fios desenvolvido pela Philips na década de 80. Desde então vem sendo muito usado para comunicação de periféricos com placas mães. O nome significa Circuito Inter-Integrado [8].

3.3.1: Estrutura

O barramento I²C consiste em duas linhas bidirecionais, uma é a Serial Data Line (SDA) e a outra é a Serial Clock (SCL). Estas linhas estão conectadas a resistores de pullup, como mostrado na Figura 3.

O endereço I²C pode conter 7 ou 10 bits, dependendo do dispositivo utilizado. A taxa de transmissão que o protocolo define é de 100kbit/s no modo padrão ou 400kbit/s no modo rápido.

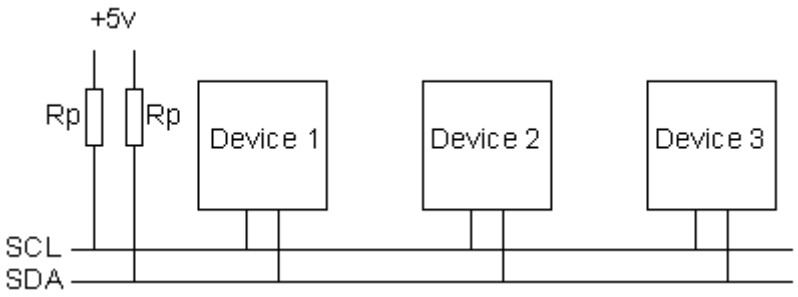


Figura 3 Estrutura Física do barramento I²C

3.3.2: Comunicação

Uma das vantagens do protocolo I²C é a simplicidade de comunicação.

A linha SDA só pode ser alterada quando a linha de clock SCL estiver em nível lógico zero. Este funcionamento só apresenta duas exceções: o start e o stop bit.

Para começar a comunicação através do barramento I²C, o mestre deve enviar um start bit e depois começar a gerar o clock de comunicação na linha SCL. O start bit nada mais é do que a mudança do nível lógico 1 para o nível lógico zero na linha de dados SDA enquanto a linha de clock SCL permanece em nível lógico 1.

O inverso ocorre no final da transmissão pelo barramento I²C, quando o mestre envia o stop bit e para de gerar o clock de comunicação. O stop bit é o inverso do start bit, ou seja, a linha de dados SDA muda de nível lógico zero para nível lógico 1 enquanto a linha de clock SCL se mantém em nível lógico 1.

O start bit e o stop bit são ilustrados na Figura 4.

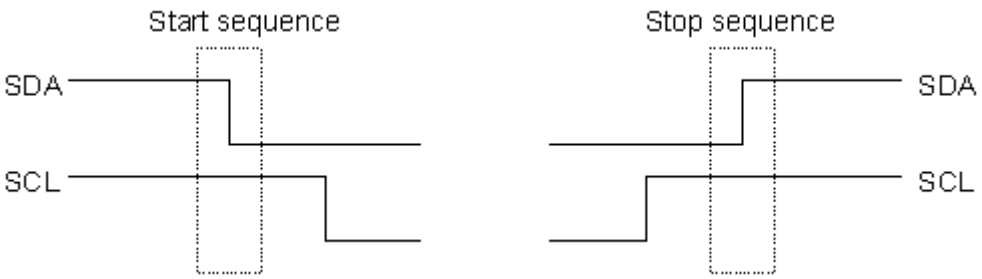


Figura 4 Start e Stop bits

Após o envio do start bit, a comunicação prossegue com o envio do endereço I²C do escravo que o mestre deseja se comunicar. Cada dispositivo possui um endereço especifico e por padrão cada endereço precisa ser único no barramento.

A comunicação acontece sempre em pacotes de oito bits (um byte) e um “ack” (alguns dispositivos permitem comunicações com tamanhos de pacotes diferenciados). O primeiro pacote carrega o endereço I²C do escravo de 7 bits e o oitavo bit informa se o acesso é uma escrita ou leitura.

Um exemplo de envio de um byte pode ser observado na Figura 5.

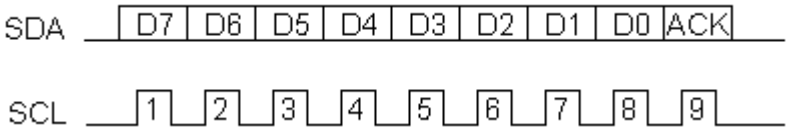


Figura 5 Byte através da I²C

Uma vez que o escravo tenha enviado o “ack”, o mestre envia um novo byte agora contendo o endereço do registrador interno ao escravo ao qual deseja acesso. Esse endereço, diferentemente do endereço I²C, possui 8 bits, já que não existe a necessidade de informar novamente se a comunicação se trata de uma escrita, ou uma leitura. De fato a primeira comunicação é sempre uma escrita, pois o mestre precisa enviar o endereço do registrador que deseja ter acesso mesmo em uma operação de leitura.

Caso o acesso represente uma escrita, o mestre começa a enviar um byte de cada vez até que finalize a transmissão com o envio do stop bit. O escravo ao receber múltiplos bytes incrementa o endereço do registrador e continua gravando os dados enviados em sequência.

Já a leitura ocorre de maneira um pouco diferenciada, pois o mestre que havia enviado um sinal de escrita para o escravo afim de enviar o endereço do registrador ao mesmo, precisa agora, enviar um novo bit de leitura. Isto é feito enviando um novo start bit sem ter enviado o stop bit. Em seguida o mestre envia novamente o endereço I^2C do escravo, mas com o oitavo bit agora sinalizando uma leitura.

A comunicação segue com o escravo mandando a informação contida no endereço especificado, e no caso de uma leitura multi-byte, continua enviando os dados contidos nos próximos registradores em sequência.

3.4: PLL

PLL (Phase Locked Loop) estão presentes em receptores de AM, FM, modems, sintetizadores de frequências, telefones sem fio, telefones celulares, instrumentos digitais e analógicos e em uma série de outras aplicações onde frequências estejam presentes. O PLL está para a freqüências assim como um amplificador operacional está para as tensões, daí sua importância na eletrônica moderna.

3.4.1: PLL Simples

A Figura 6 ilustra a arquitetura de um pll simples. Neste circuito podemos observar os blocos básicos de construção de um pll e como estão relacionados.

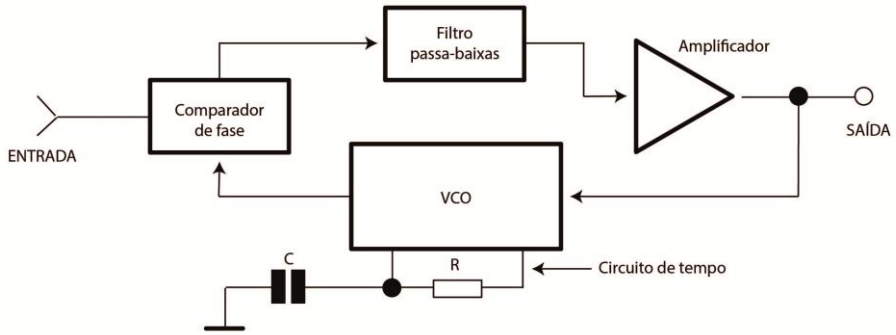


Figura 6 Arquitetura interna de um PLL simples

O bloco inicial do circuito é o “Comparador de Fase” ele funciona identificando a diferença entre a fase do sinal de entrada e a fase do sinal de saída. Este erro passa por um filtro passa-baixas. Este sinal é utilizado para o controle de um oscilador controlado por tensão ou VCO (Voltage Controlled Oscillator). Como saída, o VCO gera uma frequência que é função da tensão de entrada. Isto fecha a malha de controle interna de um pll.

3.4.2: Utilização dos PLLs

A propriedade de seguimento de referencia do pll pode ser utilizadas em uma grande gama de aplicações em que um sinal de entrada deve ser reconhecido. Uma aplicação usual é a regeneração de sinais.

Para outras aplicações em telecomunicações, as frequências de saída precisam ser exatamente iguais as de entrada. Este é o caso dos modens. Além disso, com a inclusão de um offset na fase de referência, é possível criar frequências de saída idênticas as de entrada, porém com um defasamento constante.

3.5: DCO

O DCO (digitally controlled oscillator) é um oscilador controlado digitalmente. Foi desenvolvido com o objetivo de corrigir as limitações de estabilidade dos VCOs (voltage controlled oscillator). Diferentemente dos VCO que tem a sua frequência de saída controlada pela variação da tensão de entrada, os DCO possuem entradas digitais. Isso faz com que ruídos na tensão de entrada não interfiram na frequência de saída.

3.6: GPS

O GPS ou Sistema de Posicionamento Global é um sistema baseado na navegação de satélites que informa a posição e o tempo, sobe qualquer condição climática e em qualquer lugar do mundo.

Foi desenvolvido e implementado pelo Departamento de Defesa dos Estados Unidos com fins militares e objetivo de substituir o sistema de navegação até então utilizado.

3.6.1: Funcionamento

Os receptores de GPS captam os sinais dos satélites visíveis e com base em suas informações realiza uma triangularização para determinar se posicionamento, assim como identificar o tempo exato. Para seu correto funcionamento o receptor de sinal GPS precisa “enxergar” ao menos 3 satélites, o que embora seja garantido de acordo com o Departamento de Defesa dos Estados Unidos, nem sem ocorre. Essa falha na comunicação com o satélite pode se dar por diversos fatores, como por exemplo o pouso de um pássaro na antena receptora.

3.6.2: Aplicações

Embora tenha sido originalmente desenvolvido para fins balísticos pelo governo dos Estados Unidos, atualmente com o desenvolvimento de um sistema ainda mais preciso para uso militar, o GPS foi liberado para utilização civil seguindo algumas normas de utilização. Desde sua liberação, constantemente surgem novas aplicações onde o posicionamento global é utilizado.

Como o GPS proporciona, além de informações de posicionamento, uma base de tempo precisa e confiável, muitas aplicações de sincronismo o utilizam como referência.

Capítulo 4: Especificação do Clock disciplinado

Este capítulo é dedicado em apresentar os requisitos de implementação do clock disciplinado e apresentar a arquitetura do sistema computacional utilizado para desenvolvê-lo.

4.1: Requisitos do Clock disciplinado

4.1.1: Requisitos Funcionais

O clock disciplinado deve ser capaz de gerar uma frequência que permita que os equipamentos recebam 256 amostras a cada ciclo, sendo que este ciclo corresponde ao período da frequência da rede elétrica. Além disso este sinal de clock gerado irá passar por um conversor A/D, o que faz com que a sua frequência precise ser 256 vezes maior. Sendo assim:

$$256 * 60Hz * 256 = 3.932.160Hz$$

ou

$$256 * 50Hz * 256 = 3.276.800Hz$$

Entretanto, o oscilador Si599 não é capaz de gerar frequências inferiores a 10MHz. Para contornar este problema e melhorar a precisão do clock gerado, estas frequências foram multiplicadas por 4, já que o processo de divisão do clock por 2 é extremamente simples. Assim as frequências especificadas para o funcionamento do clock disciplinado são:

- 15.278.640Hz para equipamentos operando à 60Hz
- 13.107.200Hz para equipamentos operando à 50Hz

O tempo de acomodação para que o clock disciplinado entre em regime permanente é de 60 segundos.

O clock gerado deve ser capaz de se manter a um erro inferior a 100ns por segundo. Isto significa, em outras palavras, que entre dois sinais de pps a taxa de deriva de ser menor que 100ns.

Devido a problemas de recepção de sinais do GPS existem ocasiões em que o sinal de pps desaparece, isto pode levar horas. Na ausência de um sinal pps o clock deverá se manter com a mesma configuração até que o sinal de pps volte. Com o retorno do pulso por segundo o clock disciplinado deverá se ajustar a sua frequência de funcionamento normal sem dar “saltos” de frequência. Este salto máximo possui um valor σ que pode ser ajustado para cada utilização.

O clock disciplinado deve ser capas de rejeitar perturbações no sinal de pps.

4.1.2: Requisitos não funcionais

Deve ser projetado inteiramente em lógica digital, sem auxílio de processadores e deve consumir a menor quantidade de lógica possível.

4.2: Tecnologia utilizada

Utilizando como base os requisitos do clock disciplinado e as tecnologia utilizada no projeto e construção de novos produtos na empresa, foi decidido realizar o projeto totalmente em hardware ao invés de um misto de hardware e software. Para tal, foi utilizado um FPGA (Field Programmable Gate Array). A implementação em hardware permite controlar com maior precisão um relógio externo e temos tempo de resposta muito mais curtos, além disso alguns equipamentos da empresa não possuem processador, apenas um FPGA executando funções lógicas.

Para a realização do projeto do clock disciplinado, foi utilizado um FPGA Cyclone IV E da fabricante Altera, e uma placa desenvolvida na empresa que

contém o circuito necessário para a comunicação com o oscilador Si599 desenvolvido pela Silicon Laboratories.. O clock disciplinado é, portanto, um circuito lógico digital.

Por ser implementado em FPGA e por ser um circuito lógico digital, alguns outros requisitos não funcionais são adicionados:

- O Clock disciplinado como circuito digital bem construído, deve possuir atrasos de propagação dentro dos limites do clock utilizado de 50 MHz, e respeitar os requisitos temporais de funcionamento dos elementos de memória (flip-flops basicamente).
- O clock disciplinado deve respeitar os limites de elementos lógicos e memória disponíveis no FPGA.

Para desenvolvimento do trabalho, utilizou-se um kit de desenvolvimento que além do FPGA, possui LEDs, chaves e botões, oscilador de 50Mhz, circuito I²C embarcado, interface de uso genérico GPIO, entrada de clock externo e outros elementos que não serão citados, pois não foram utilizados nesse trabalho.

4.2.1: FPGA

A lógica digital [9] faz a transformação de valores que presentam grandezas reais em um sistema de numeração de base 10 para um sistema de numeração de base 2, onde só existem os valores lógico 0 e 1. Isto faz com que circuitos lógicos digitais possam ser analisados e projetados através dos meios usuais para realizar operações em sistemas binários, como por exemplo, lógica booleana.

Existem dois tipos de circuitos lógicos. Quando em um circuito lógico digital a suas saídas são determinadas exclusivamente por suas entradas, dizemos que este circuito é combinacional. Qualquer circuito combinacional pode ser construído à partir das funções lógicas AND, OR e NOT.

Já quando um circuito depende não somente de suas entradas atuais, mas também de suas entradas passadas, dizemos que este é um circuito sequencial. Circuitos sequenciais possuem memória, isso faz com que estes circuitos sejam constituídos por Flip_flops e Latches além das portas lógicas presentes nos circuitos

combinacionais. A diferença entre flip-flops e latches é que no caso dos flip-flops, sua saída é atualizada conforme as bordas de subida de um sinal de clock. Já os latches atualizam a sua saída sem levar em consideração o sinal de clock.

Um FPGA é um circuito integrado desenvolvido para ser configurado pelo cliente ou desenvolvedor após produção [10].

Internamente a um FPGA existem os “blocos lógicos” que são interconectados através de uma hierarquia bem definida. A configuração destes blocos pode gerar teoricamente qualquer circuito combinacional, se forem respeitados os limites de blocos disponíveis. Quase sempre, estes blocos também apresentam elementos de memória, possibilitando assim a construção de qualquer circuito sequencial, ou em outras palavras qualquer circuito lógico.

A grande vantagem de um FPGA é a sua possibilidade de reconfiguração, o que geralmente é feito através de uma linguagem de descrição de hardware (HDL). Com isso, um circuito pode ser projetado e corrigido sem que nenhum circuito real seja produzido, diminuindo assim os custos de produção e o tempo necessário para o desenvolvimento.

4.2.2: O kit de desenvolvimento

Para desenvolver o clock disciplinado foi utilizado o kit de desenvolvimento DE2-115 da Terasic, mostrado na Figura 7. Esse kit de desenvolvimento foi escolhido para o projeto por ser o kit disponível na empresa e por atender as necessidades do projeto.

Desse kit de desenvolvimento foi utilizado o FPGA cyclone IV E, a porta de comunicação genérica GPIO, os LEDs, e os botões.

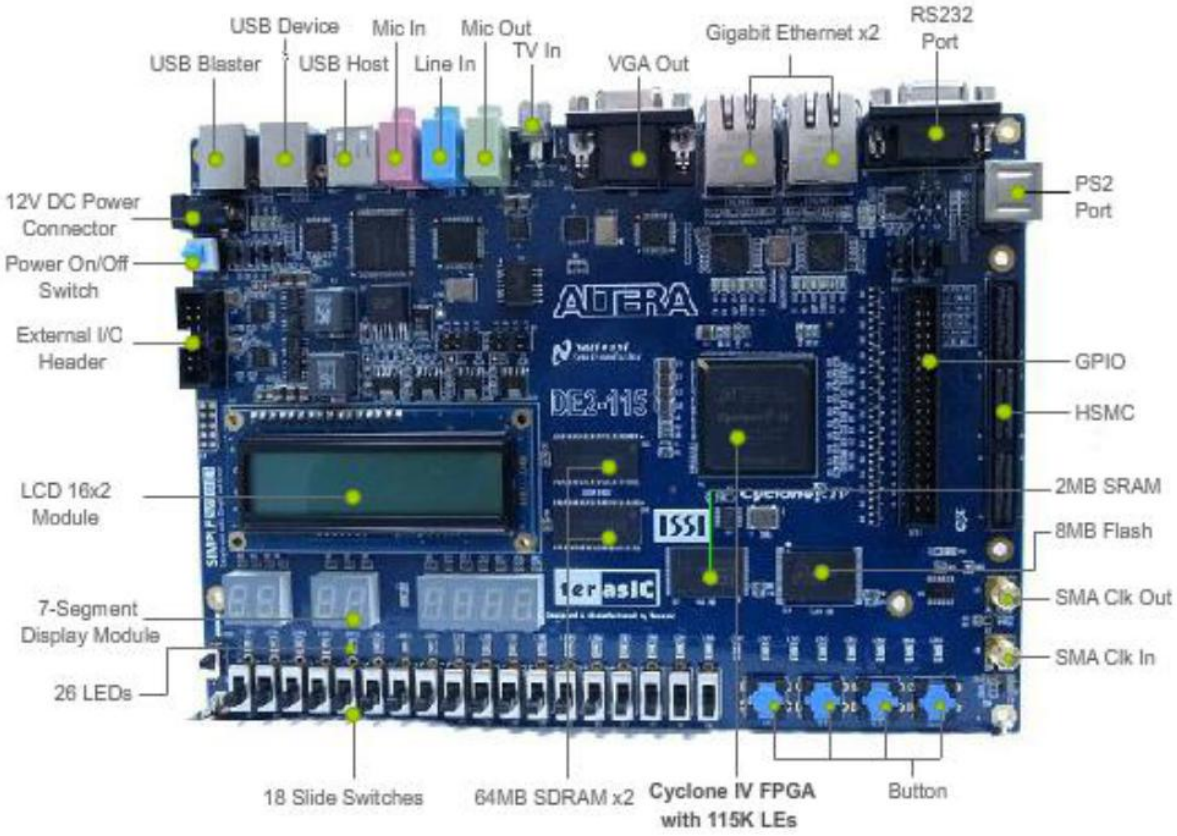


Figura 7 Kit de desenvolvimento DE2-115

4.3: Conclusão

Este capítulo listou os requisitos de implementação do clock disciplinado, os quais se basearam nas necessidades da empresa e também na tecnologia escolhida para criar o clock disciplinado, também apresentada nesse capítulo.

No próximo capítulo será mostrado como o clock disciplinado foi projetado nesse trabalho, apresentando a estrutura proposta e o funcionamento de cada componente do circuito lógico digital.

Capítulo 5: Projeto do Clock Disciplinado.

Nesse capítulo será apresentado, em detalhes, o projeto do circuito digital implementado em um FPGA. Diagramas são utilizados para apresentar os módulos do circuito digital assim como o relacionamento entre eles, enquanto máquinas de estados são utilizadas para detalhar o funcionamento de cada componente do projeto.

As máquinas de estado são apresentadas de maneira mais simplificada, omitindo alguns detalhes de implementação e focando mais na funcionalidade, com o objetivo de facilitar a compreensão.

5.1: Arquitetura do Clock Disciplinado

Após a compreensão do problema, da especificação dos requisitos do projeto e da escolha da tecnologia utilizada (FPGA), a próxima etapa do trabalho é projetar o circuito digital.

Para atender os requisitos do projeto e para realizar as funcionalidades necessárias para implementação de um gerador de clock disciplinado, foi proposto nesse trabalho que o circuito digital tenha a estrutura mostrada na Figura 8.

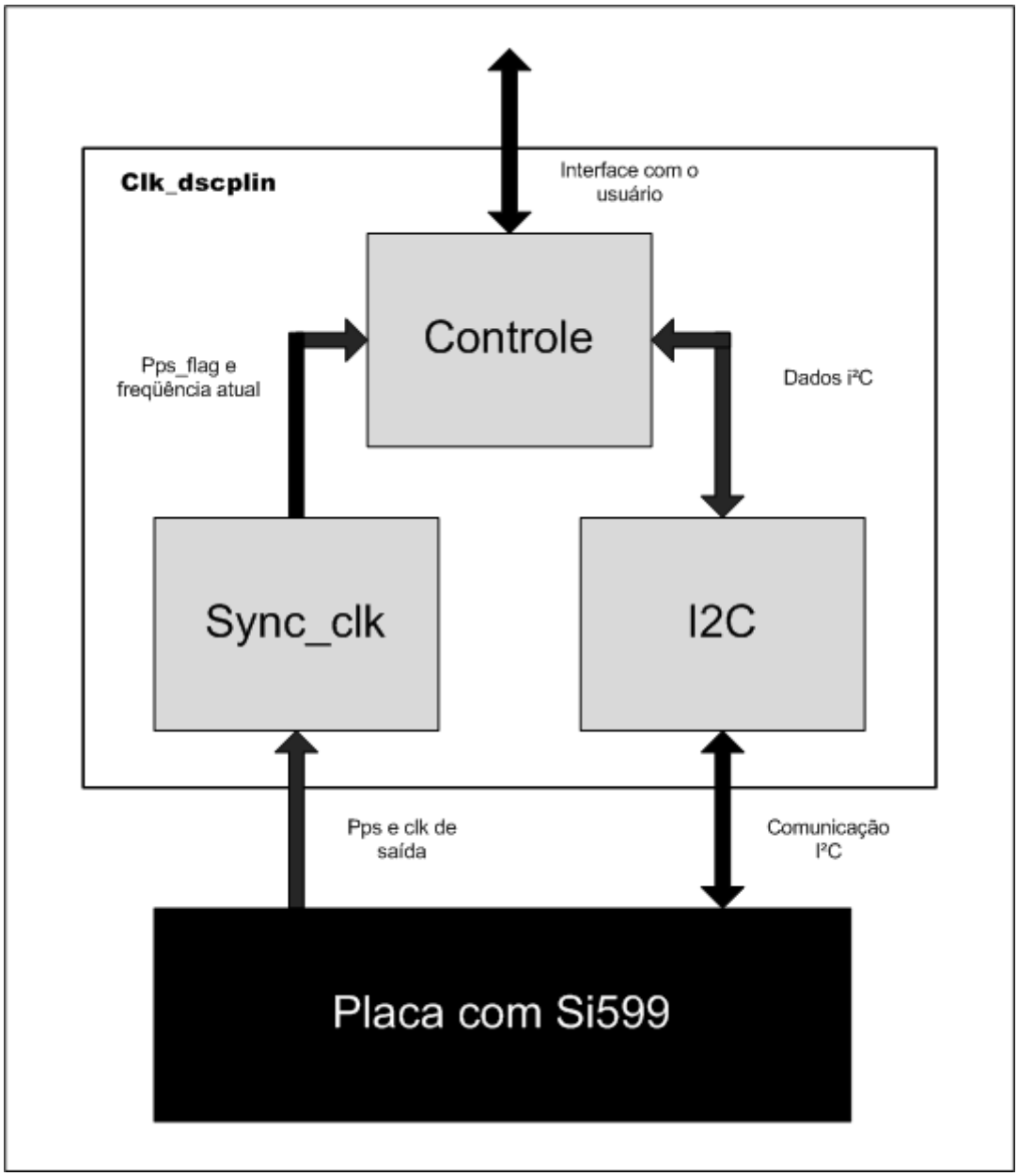


Figura 8 Estrutura do Clock Disciplinado

O clock disciplinado é formado por 3 componentes principais (cada componente é um circuito lógico digital) representados pelos blocos cinzas do diagrama:Controle, Sync_clk e I2C. Esses componentes são conectados como representado na Figura 8 e também se comunicam com uma placa externa onde se encontra o oscilador Si599 representada pelo bloco preto.

As setas indicam as conexões entre os módulos, e o sentido delas indica o sentido do fluxo de dados. O módulo receptor dos dados ou que está sendo

acessado gera sinais de resposta durante as transações para o módulo transmissor, para que este possa saber os resultados de uma operação e quando ela foi finalizada. Esses sinais de resposta também são representados no diagrama.

O bloco “Controle” é responsável por comandar as trocas de dados do clock disciplinado e controlar o clock de saída do sistema. Ele recebe um sinalizador de entrada de pps e o valor da frequência atual do clock de saída do módulo “Sync_clk”, ele também se comunica com o módulo “I2C” para ler e escrever dados nos registradores do Si599. Este bloco também é responsável pelo acionamento de leds que indicam as condições de funcionamento para o usuário.

O bloco “Sync_clk” é responsável pela aquisição dos dados vindos da placa. Ele recebe o sinal do pps externo e o sinal de clock gerado pelo Si599 e envia a informação para o “Controle”.

O bloco “I2C” é responsável por toda a comunicação I²C entre o FPGA e a placa. Ele recebe as requisições do “Controle”, se comunica com a placa através do protocolo I²C e retorna os dados.

Nas seções 5.2, 5.3, 5.4 será apresentada a estrutura interna de “Controle”, “Sync_clk” e “I2C”.

5.2: Controle

A Figura 9 mostra a estrutura do “Controle”. O “Cotrole” é formado por 2 componentes: “Calculate_large” e “Calculate_small”. Este módulo também possui uma logica interna que é representado na Figura 10.

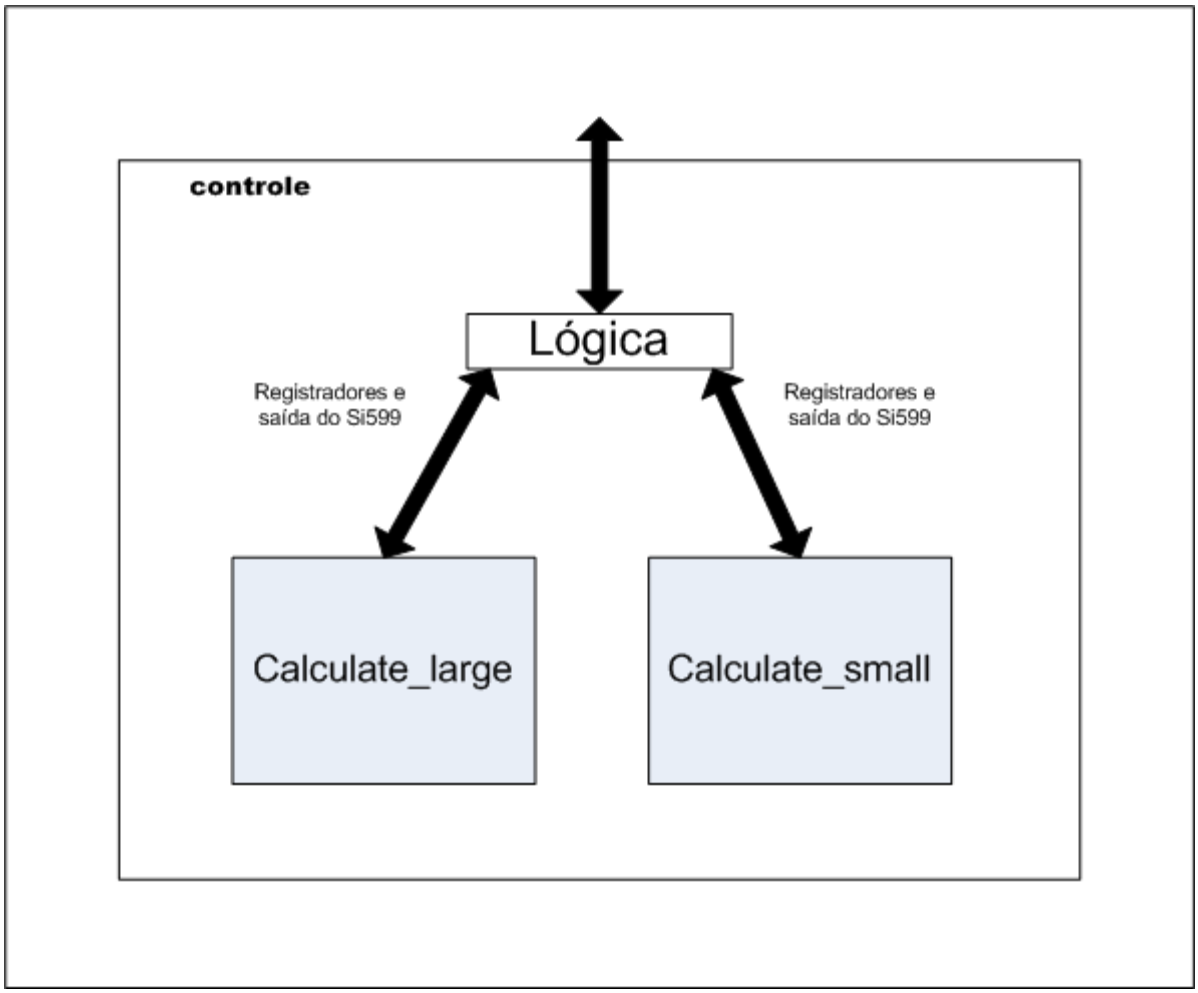


Figura 9 Módulo de Controle

5.2.1: Lógica do Controle

A máquina de estados do “Controle” é exibida na Figura 10. Este módulo atua como o controle da lógica desenvolvida e também atua como controlador do clock de saída. Sempre que um novo pps é sinalizado a lógica começa um novo ciclo. Exitem dois tipos de ajustes a serem realizados, o ajuste para grandes modificações no clock de saída (“Large”) e o ajuste para pequenas modificações no clock de saída (“Small”). O ajuste para grandes modificações é mais robusto porém só é realizado no momento da inicialização do funcionamento do clock disciplinado, uma vez que é necessário congelar o DCO para realizar esse ajuste. Durante a operação normal do sistema apenas o ajuste fino é realizado.

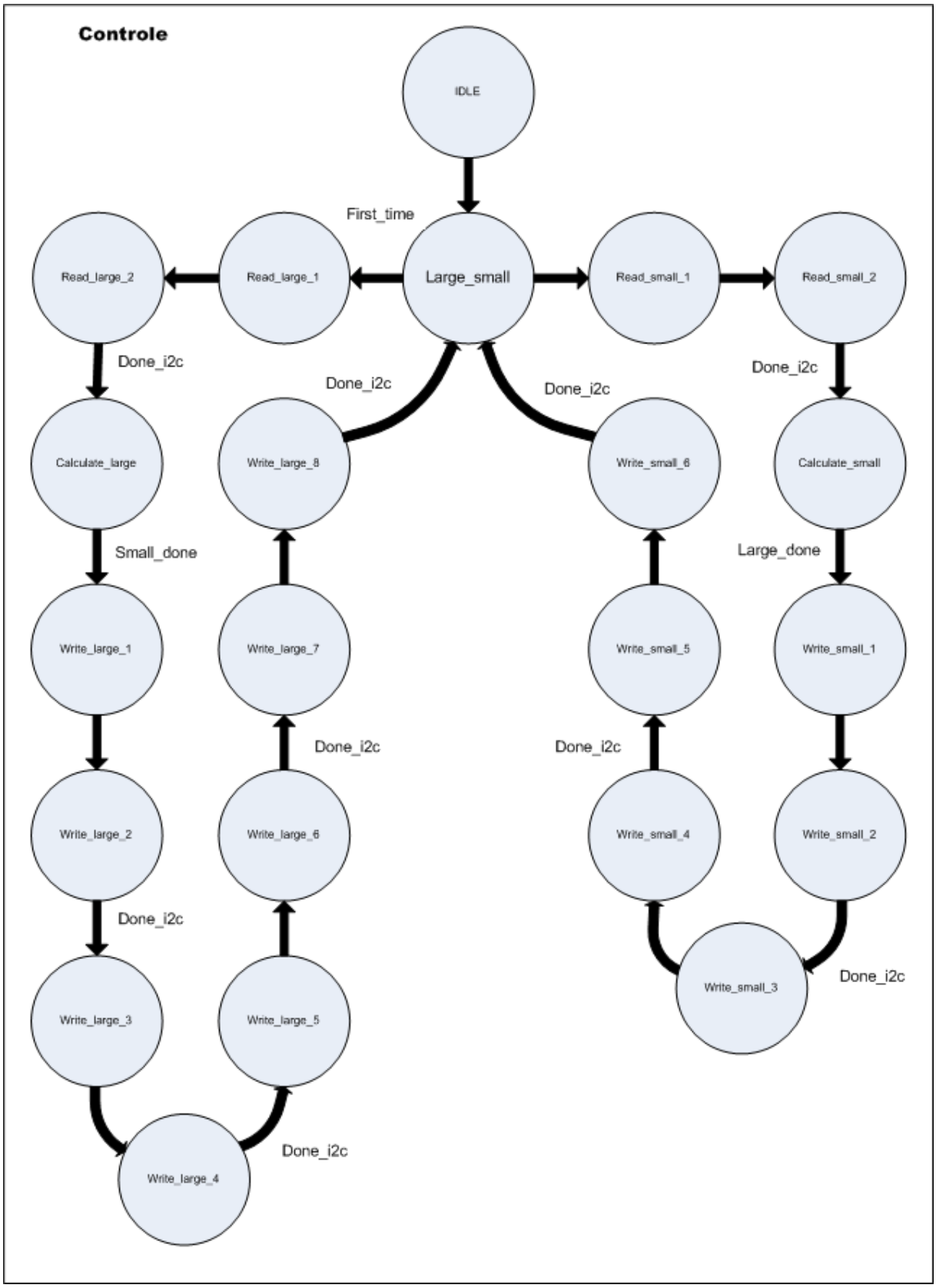


Figura 10 Máquina de estados simplificada do Controle

5.2.1.1: Ciclo “Large”

Este ciclo é realizado para que a frequência de saída do clock disciplinado fique próxima o suficiente da referência de modo que o ajuste fino realizado no ciclo “Small” consiga manter a frequência dentro dos padrões desejados.

Os estados iniciais “Read_large_1” e “Read_large_2” enviam as ordens de leitura e aguardam a confirmação da operação de leitura realizada com sucesso, respectivamente.

Em seguida, no estado “Calculate_large” os valores dos registradores e a frequência gerada pelo Si599 são passados para o bloco com o mesmo nome que faz as contas necessárias para gerar os novos valores dos registradores que são devolvidos ao módulo “Controle” para futura gravação no oscilador.

Com os novos valores obtidos, as ordens de gravação dos registradores são enviadas nos estados seguintes: “Write_large_1”, “Write_large_2”, “Write_large_3”, “Write_large_4”, “Write_large_5”, “Write_large_6”, “Write_large_7” e “Write_large_8”.

O procedimento de escrita dos dados é realizado conforme explicado no capitulo 6.2.2, onde “Write_large_1” e “Write_large_2” enviam os dados para congelamento do DCO. Com o DCO congelado os estados “Write_large_3” e “Write_large_4” enviam os novos valores dos registradores “rfreq”, “hsdiv” e “n1” para serem gravados.

Após o gravamento dos novos valores dos registradores o procedimento continua com a liberação (descongelamento) do DCO, o que é realizado nos estados “Write_large_5” e “Write_large_6”. Por último existe a necessidade de setar o bit “new_freq_bit” de um registrador específico que sinaliza ao Si599 que novos valores de frequência e divisores foram inseridos. Este procedimento funciona como qualquer outra escrita I^2C e é realizado nos estados “Write_large_7” e “Write_large_8”.

Após o ciclo ter terminado a máquina de estados volta ao estado “Large_small” onde fica aguardando uma nova sinalização de recebimento de pps.

5.2.1.2: Calculate_large

A Figura 11 representa a máquina de estados simplificada do módulo Calculate_large.

Este recebe os valores atuais dos registradores “rfreq”, “hsdiv” e “n1”, assim como recebe o valor da frequência atual de saída do Si599. Com essas informações ele realiza os cálculos descritos no capítulo 6.2.4 para grandes ajustes de frequência do oscilador.

A necessidade de criar uma máquina de estados para a realização de uma sequência de contas provém da necessidade da realização de divisões que são muito dispendiosas e consomem muita lógica. Como o período de amostragem do clock de saída é de 1 segundo e o restante do processamento é da ordem de milissegundos, foi possível utilizar um algoritmo otimizado de divisão que consome menos lógica, porém é mais lento.

Isso faz com que condições precisem ser adicionadas para que as operações matemáticas só sejam realizadas quando os valores de cada etapa estiverem corretos e estáveis.

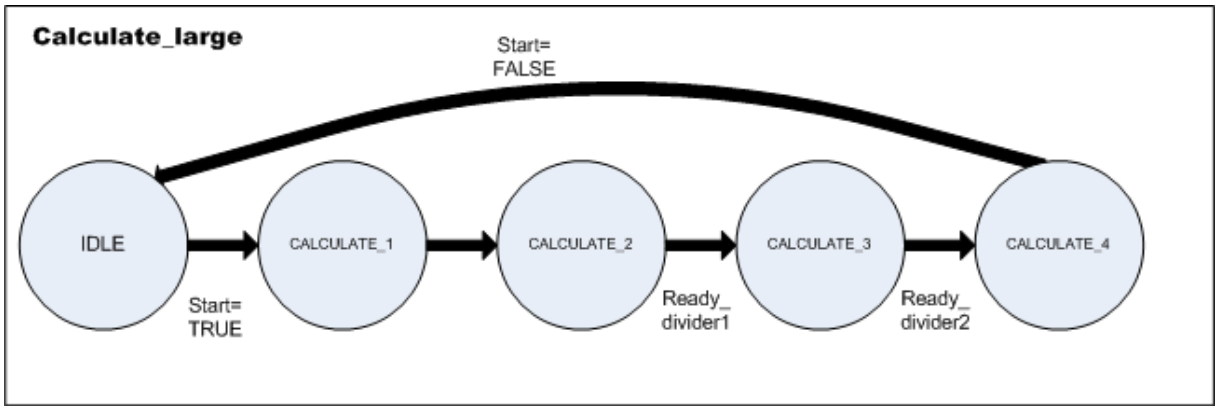


Figura 11 Máquina de estados simplificada do módulo Calculate_large

5.2.1.3: Ciclo “Small”

Ao que se refere a lógica da máquina de estados do ciclo “Small”, ela é quase idêntica ao ciclo “Large”, porém como neste ciclo não são alterados os valores dos registradores “hsdiv” e “n1” os cálculos e procedimentos de gravação diferenciam-se ligeiramente.

Os estados iniciais “Read_small_1” e “Read_small_2” enviam as ordens de leitura e aguardam a confirmação da operação de leitura realizada com sucesso, respectivamente.

Em seguida, no estado “Calculate_small” os valores dos registradores e a frequência gerada pelo Si599 são passados para o bloco com o mesmo nome que faz as contas necessárias para gerar os novos valores dos registradores que são devolvidos ao módulo “Controle” para futura gravação no oscilador.

Com os novos valores obtidos, as ordens de gravação dos registradores são enviadas nos estados seguintes: “Write_small_1”, “Write_small_2”, “Write_small_3”, “Write_small_4”, “Write_small_5” e “Write_small_6”.

O procedimento de escrita dos dados é realizado conforme explicado no capitulo 6.2.3, onde “Write_small_1” e “Write_small_2” enviam os dados para congelamento do bit M. Com o bit M congelado os estados “Write_small_3” e “Write_small_4” enviam o novo valor do registrador “rfreq” para ser gravado. Após o gravamento do novo valor do registrador o procedimento continua com a liberação (descongelamento) do bit M, o que é realizado nos estados “Write_small_5” e “Write_small_6”.

Após o ciclo ter terminado a máquina de estados volta ao estado “Large_small” onde fica aguardando uma nova sinalização de recebimento de pps.

5.2.1.4: Calculate_small

A Figura 12 representa a máquina de estados simplificada do módulo Calculate_small.

Este recebe o valor atual do registrador “rfreq”, assim como recebe o valor da frequência atual de saída do Si599. Com essas informações ele realiza os cálculos descritos no capítulo 6.2.3 para pequenos ajustes de frequência do oscilador.

Como já descrito para o módulo “Calculate_large”, existe a necessidade de criar uma máquina de estados para a realização de uma sequência de contas provém da necessidade da realização de divisões que são muito dispendiosas e consomem muita lógica. Como o período de amostragem do clock de saída é de 1 segundo e o restante do processamento é da ordem de milissegundos, foi possível utilizar um algoritmo otimizado de divisão que consome menos lógica, porém é mais lento.

Isso faz com que condições precisem ser adicionadas para que as operações matemáticas só sejam realizadas quando os valores de cada etapa estiverem corretos e estáveis.

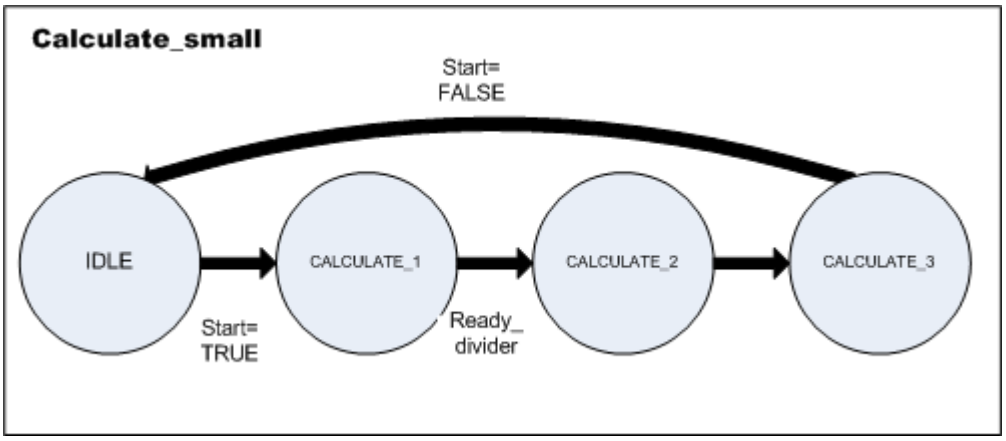


Figura 12 Máquina de estados do módulo Calculate_small

5.3: I2C

A Figura 13 mostra a máquina de estados simplificada do módulo I2C desenvolvido. Apesar do seu maior número de estados, seu comportamento sequencial a torna de fácil compreensão.

Neste módulo foi implementado o Mestre do protocolo I²C, uma vez que o Si599 é o escravo na comunicação.

De acordo com o protocolo de comunicação I²C, para se estabelecer a comunicação entre mestre e escravo o mestre precisa, além de transmitir os dados, gerar um clock de referencia para a comunicação comunicação. Para isso o clock principal do FPGA de 50MHz passa pelo bloco divisor de clock “clk_divider” que gera um clock de referencia interno ao módulo de 800kHz. Este clock precisa ser no mínimo duas vezes mais rápido que o clock de comunicação I²C para que se tenha um controle preciso sobre as duas linhas de comunicação do protocolo I²C. Esta comunicação está detalhada no capítulo 6.2.5.

Com a necessidade de realizar tarefas diferentes quando o barramento de clock I²C está em alto e baixo, a gravação ou leitura de blocos de 8 bit é realizadas em loops de dois estados e não de um. Isso serve para que um gere o sinal alto no barramento de clock e o outro o sinal baixo e isso se repete durante toda transmissão de cada bloco de 8 bits. Este comportamento pode ser observado nos estados: “Start_2” e “Start_3”, “Address_1” e “Address_2”, “Write_1” e “Write_2”, “Restart_address_1” e “Restart_address_2”, e “Read_1” e “Read_2”.

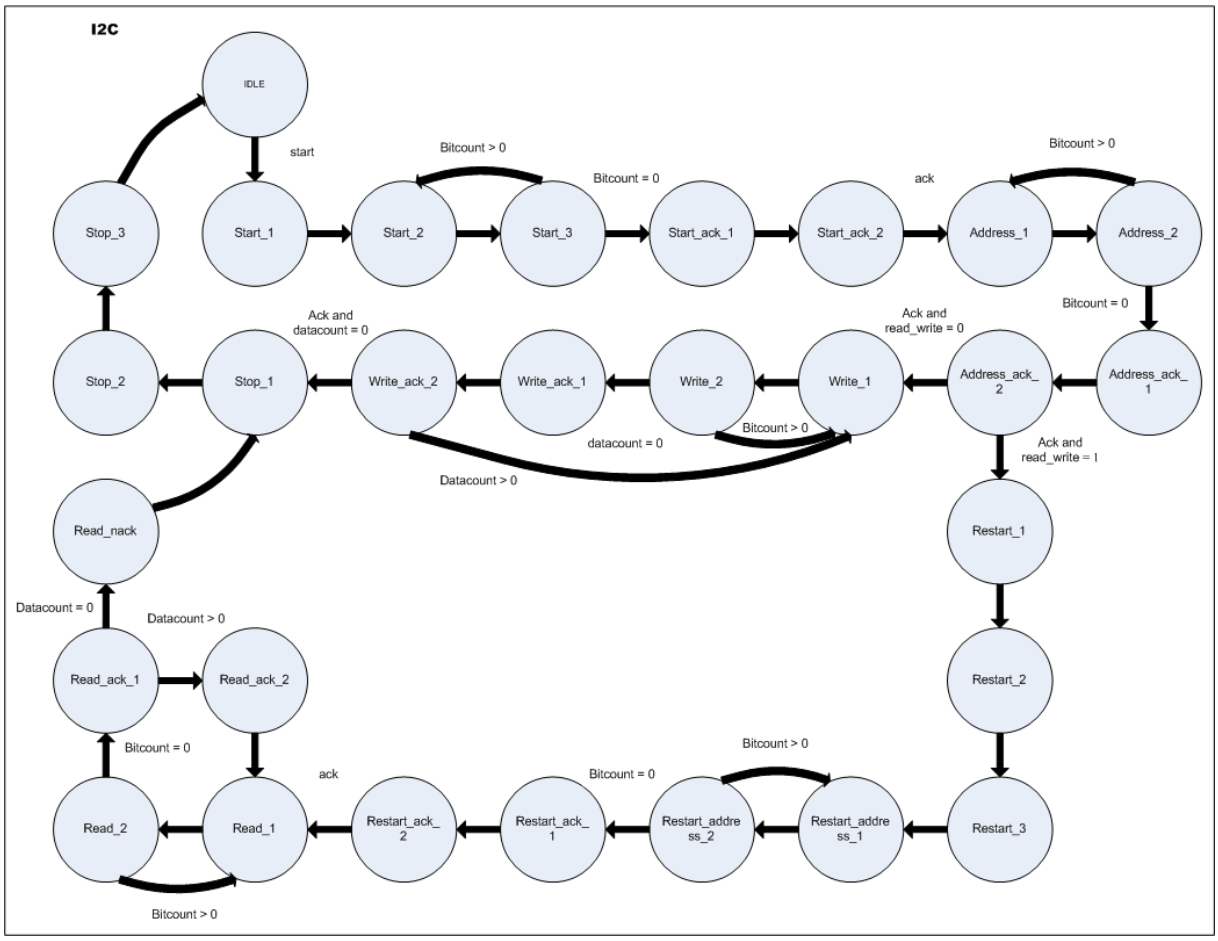


Figura 13 Máquina de estados do módulo I²C

Os estados “Start_1”, “Start_2” e “Start_3” são responsáveis pelo começo do protocolo de comunicação, enviando o “start bit” e em seguida o endereço I2C do escravo. Caso o escravo esteja escutando ele reponde com um sinal de um bit de “ack”. Este sinal é captado nos estados “Start_ack_1” e “Start_ack_2” e com isso uma nova etapa na comunicação é iniciada.

Caso o sinal de ack não seja recebido todo o processo recomeça, voltando para o estado “Start_1”. Esta é uma medida de garantia da consistência dos dados que ocorre em todos estados onde há recebimento de “akcs” e para deixar a figura mais limpa esses caminhos não foram incluídos.

Na sequência do protocolo de comunicação os estados “Address_1” e “Address_2” enviam o endereço do registrador as ser lido ou escrito e os estados

“Address_ack_1” e “Address_ack_2” recebem o “ack” do escravo ou, em caso negativo, retornam ao estado “Start_1” como descrito anteriormente.

Como os procedimentos iniciais de escrita e leitura são idênticos no protocolo I²C a máquina de estados só diverge no estado “Address_ack_2” onde o próximo estado pode ser “Write_1” ou “Restart_1” dependendo se o módulo estiver realizando uma escrita ou leitura respectivamente.

No caso de uma escrita, os estados “Write_1” e “Write_2” escrevem os oito bits de dados da mesma forma que nos estados “Address” e “Start” com uma única diferença. Ao contrario dos blocos de endereço que são sempre únicos, os blocos de dados podem ser múltiplos e sendo assim uma série de escritas consecutivas precisa ser realizada.

Após o recebimento do “ack” existe uma condição que verifica se mais dados precisam ser escritos, em caso afirmativo, a máquina de estados volta para o estado “Write_1” quantas vezes for necessário para a conclusão da escrita. Mais uma vez o não recebimento de algum “ack” implica no recomeço de todo o processo no estado “Start_1”.

Com a conclusão da escrita os estados “Stop_1”, “Stop_2” e “Stop_3” se encarregam de enviar o “stop bit” para o escravo e com isso encerrar a comunicação.

No caso da escrita um novo “start bit” deve ser enviado para que se proceda à leitura. Os estados “Restart_1”, “Restart_2” e “Restart_3” enviam esse novo “start bit” e os estados “Restart_address_1” e “Restart_address_2” reenviam o endereço I²C do escravo como descreve o protocolo. Mesmo com grande semelhança com os estados “Start” originais, foi optado deixar esta etapa da comunicação em estados independentes devido a particularidades desse novo “start bit” e problemas no desenvolvimento do projeto, contudo é possível realizar uma otimização futura que junte os dois procedimentos nos mesmos estados.

Os estados “Restart_ack_1” e “Restart_ack_2” recebem o “ack” do escravo ou retornam para o estado “Start_1” caso não recebam nada.

Os estados “Read_1” e “Read_2” funcionam de maneira semelhante aos estados “Write_1” e “Write_2” e são executados enquanto houverem dados a serem

lidos. Já o estado “Read_ack_1” é diferente pois agora quem envia o “ack” é o mestre e não o escravo. Caso ainda existam dados a serem lidos o estado “Read_ack_2” é necessário apenas para manter o barramento de clock funcionando corretamente antes de voltar ao estado ”Read_1”.

Se a leitura já estiver sido concluída o estado “Read_nack” envia o sinal “not ack” ao escravo como descrito no procedimento de finalização da leitura I²C e em seguida os estados “Stop_1”, “Stop_2” e “Stop_3” enviam o “stop bit” e encerram a comunicação.

5.4: Sync_clk

O módulo “Sync_clk” é responsável pela aquisição dos dados de saída, assim como pela sincronização dos mesmos junto ao pps e ao clock interno do FPGA.

Este módulo é composto por 4 blocos: “pps_filter”, “edge_detector”, “alt_pll_megafunction” e “pll_counter”, e está representado na Figura 14.

Como entrada ele recebe o sinal do clock de saída do clock disciplinado e o pulso pps vindo de um receptor gps.

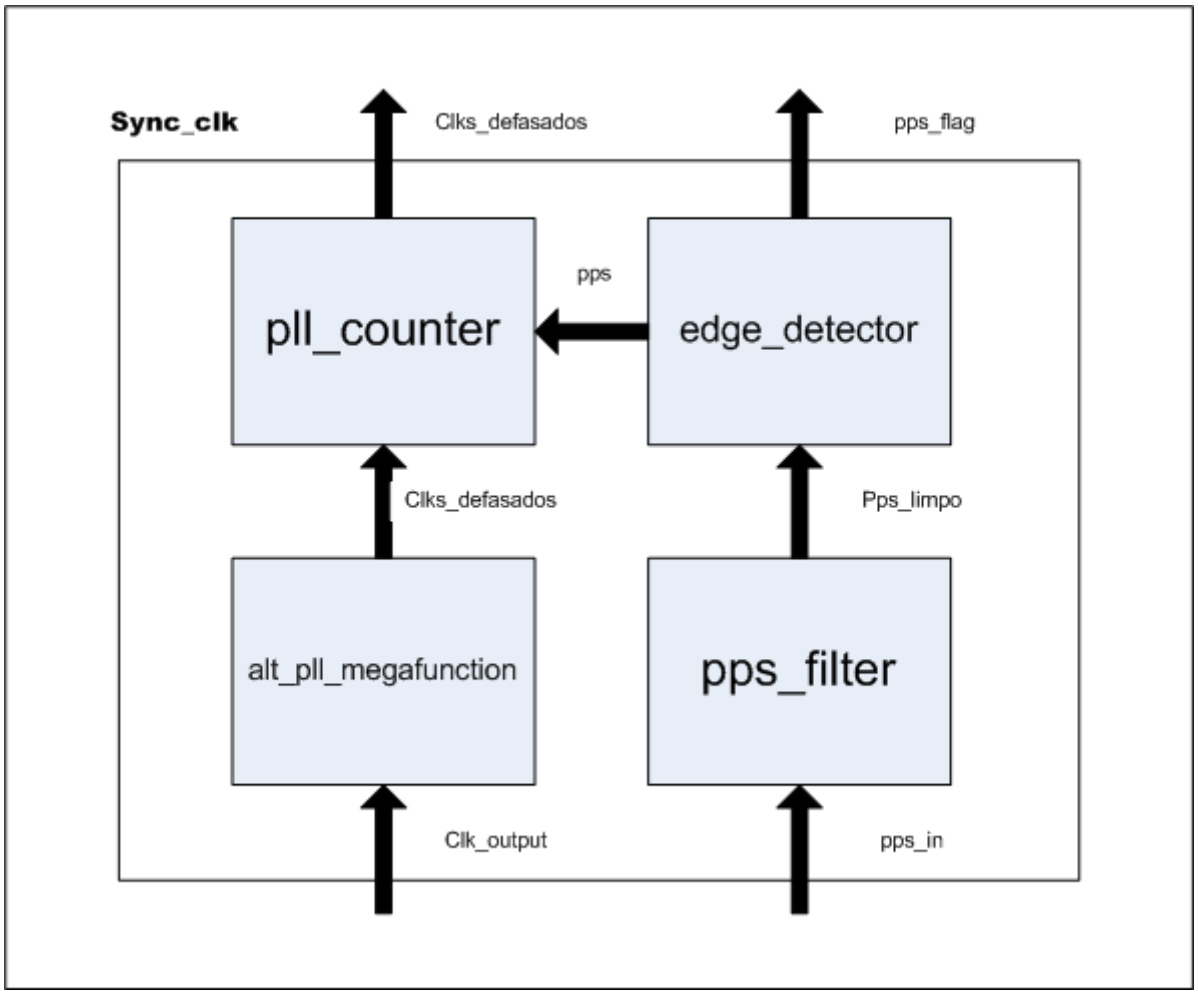


Figura 14 Estrutura do módulo Sync_clk

5.4.1: Pps_filter

Devido a ruídos presentes no sinal do pps, surgiu a necessidade de se incluir um filtro de sinal para eliminar as oscilações do pulso. Isso só foi necessário pois o pulso de pps tem uma duração suficiente para que ruídos sejam interpretados como novas bordas de subida no mesmo pulso, o que causa problemas de contagem entre outros.

Este módulo simplesmente reconhece a primeira borda de subida do pps e como saída gera uma onda quadrada de duração menor que 1 segundo para eliminar completamente os problemas de ruídos no sinal recebido.

5.4.2: Edge_detector

Este bloco também é muito simples pois realiza uma única função. Como entrada ele possui qualquer forma de onda digital, e gera como saída um pulso do tamanho de um ciclo de clock interno (no caso 20ns) a cada borda de subida.

Usado junto a registradores este módulo garante que sua saída seja um pulso de um ciclo sincronizado com o clock interno, o que é essencial quando se tem mais de um clock em um mesmo sistema, para que não haja problemas de sincronização.

Por esse motivo esse bloco além de ser usado na entrada do pps, também é usado internamente ao bloco “pll_counter” para fazer a sincronização dos clocks defasados.

5.4.3: Alt_pll_megafunction

Este bloco é gerado automaticamente pela ferramenta “Megafunction Wizard” do software Quartus 2 da Altera. Ele é necessário para que se possa usar um dos 4 plls presentes no kit de desenvolvimento.

Como entrada ele recebe um sinal de clock diferencial e como saída gera 4 sinais de clock. O primeiro é idêntico à entrada, o segundo é defasado em 45 graus, o terceiro em 90 graus e o quarto em 135 graus. Todos os clocks de saída possuem a mesma frequência e só diferem em suas fases por 45 graus.

Esta metodologia é utilizada para melhorar o erro na contagem das bordas de subida no clock gerado de um período para um oitavo de período. Este resultado é obtido não apensa defasando o clock a ser medido 3 vezes em 45 graus, mas também utilizando cada um desses novos clocks defasados juntamente com o sem defasagem em suas versões negadas. Negar um clock insere uma fase de 180 graus e fazendo isso em todos os clocks defasados é possível obter um espectro de oito clocks defasados a cada 45 graus espalhados igualmente por 360 graus.

5.4.4: pll_counter

Este bloco é responsável pela contagem da frequência de saída do clock disciplinado. Cada um dos 4 clocks de entrada são negados e os 8 são enviados para “edge_detectors” após terem passado por registradores como explicado anteriormente no módulo “edge_detector”. Seu funcionamento está representado na Figura 15.

Na saída de cada “edge_detector” sai um pulso com uma duração de 20ns para cada borda de subida. Estes pulsos são contados pelos blocos “frequency_meter” entre cada pulso de pps. Os resultados de cada contagem individual voltam para a lógica do módulo e são somados em árvore para que se tenha a contagem total de todas as bordas de subida dos 8 clocks defasados.

Com essa abordagem um erro de 45 graus em um período ao final de um segundo já seria suficiente para causar um erro com relação a referência, isto não ocorreria caso essa divisão do clock não fosse feita e seria necessário que a frequência variasse um período completo para que fosse percebida a diferença com relação a referência.

Fazendo um breve comparativo com uma malha de controle, isto seria equivalente a multiplicar por 8 a precisão do sensor.

5.4.4.1: Frequency_meter

Este bloco é responsável por contar os pulsos de clock de um clock específico e como se trata de um simples contador, não foi ilustrado por figuras.

Como entrada possui o clock a ser contado e um sinalizador de pps. Este módulo possui um funcionamento simples e nada mais é que um contador de pulsos que zera a cada pulso de pps e tem como saída o resultado da contagem de pulsos. O resultado é mantido em um registrador para que a saída seja sempre constante ao longo da contagem.

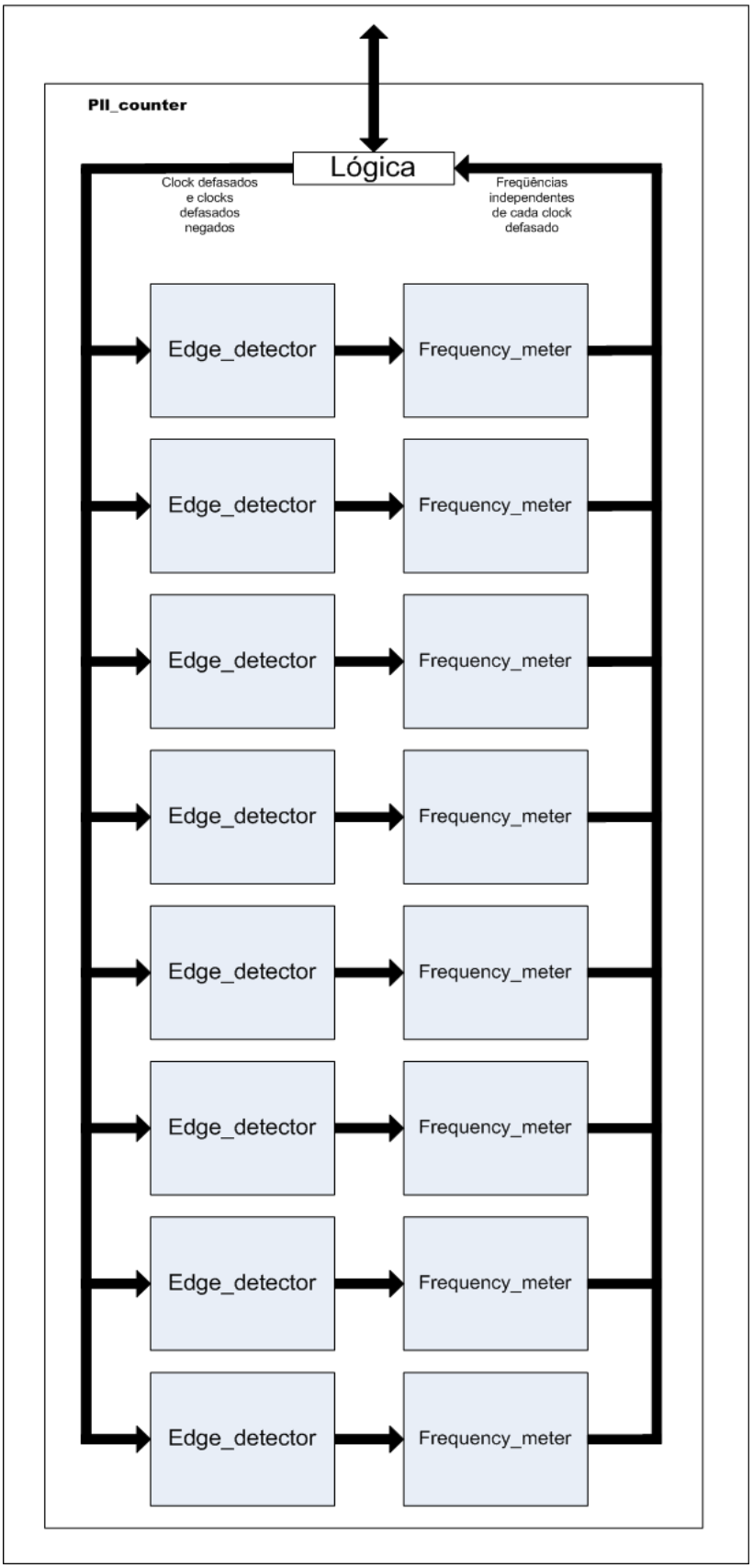


Figura 15 Estrutura do módulo Pll_counter

5.5: Conclusão

Esse capítulo apresentou o projeto detalhado do circuito digital do clock disciplinado. Este circuito foi implementado no FPGA do kit de desenvolvimento, completando a arquitetura descrita no capítulo 4.

No próximo capítulo será mostrado como o FPGA foi programado para implementar o projeto apresentado neste capítulo, assim como os testes realizados e os resultados obtidos.

Capítulo 6: Implementação, Testes e Resultados

Nesse capítulo é explicado como o projeto apresentado no capítulo 5 foi realizado em circuito digital, apresentando a linguagem VHDL, assim como as ferramentas utilizadas para a programação do FPGA.

Durante esse capítulo também serão mostrados as ferramentas utilizadas para testes, os testes realizados e os resultados obtidos.

6.1: Máquinas de estados síncronas

As máquinas de estado citadas no capítulo anterior trabalham em circuito lógico digital como máquinas de estados síncronas, que são circuitos sequenciais com elementos flip-flops de memória, onde é utilizado um mesmo sinal de clock. Outra característica das Máquinas de estado é a sua capacidade de modificar seu estado, no caso a saída dos flip-flops, de uma única maneira, com a mudança do sinal dos clocks de zero para um ou o contrário.

Quando falamos da memória de estado, temos um conjunto de n flip-flops que armazenam o atual estado da máquina, que no caso são 2n estados distintos um do outro, o próximo estado da máquina consiste na lógica do próximo estado F, que é função do estado atual e das entradas. A determinação da lógica de saída G é feita geralmente, em função do estado atual, mas pode ser função da entrada em alguns casos. Os estados F e G são considerados circuitos combinacionais conforme ilustrado na Figura 16.

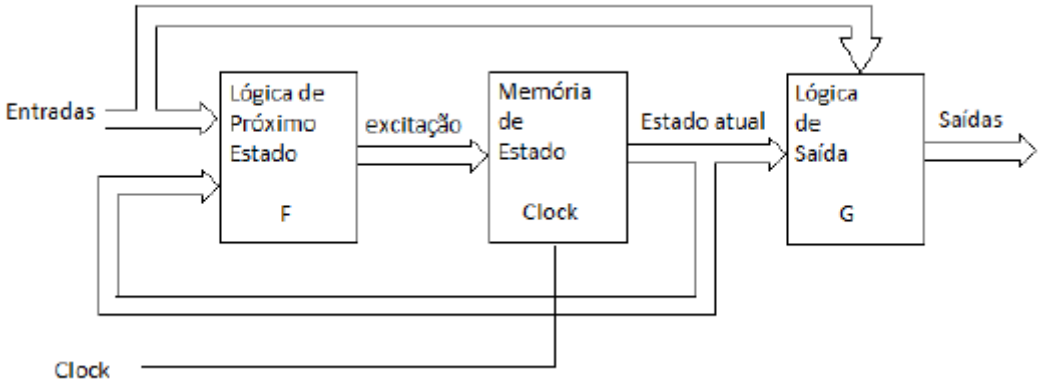


Figura 16 Estrutura de uma máquina de estados síncrona

Em vista da representação das máquinas de estados do capitulo anterior para as de estado síncronas, os estados estão representados pela memória de estado, enquanto os eventos de transição dos estados compõem a lógica do próximo estado. Já os sinais de saída dos componetes estão fazendo parte da lógica de saída.

6.2: VHDL

Uma abstração corresponde a um modelo simplicado do sistema, em se tratando de circuitos digitais, elencando apenas parte das características e excluindo detalhes associados, isso com o objetivo de minimizar o volume de erros, dando destaque as informações com maior importância. Sendo assim classifica-se uma abstração de alto nível aquela cujo contenha apenas as informações relevantes, e uma abstração de baixo nível o tipo mais preciso, mais complexo e mais próximo ao circuito real cujo contenha também as informações ignoradas no exemplo anterior. Quando se desenvolvem circuitos digitais, a primeira etapa consiste em uma abstração de mais alto nível, porém com o passar do tempo são implementadas melhorias e o sistema é mais bem compreendido até se chegar em um circuito real.

HDL é a linguagem modelada para um hardware, que tem como objetivo descrever precisamente um circuito digital no nível da abstração que se está desejando, uma característica marcante é que sua semântica e uso diferem muito das demais linguagens tradicionais, assim como C, por exemplo, que é modelada em um processo sequencial. Facilitando o desenvolvimento do algorítimo em cada etapa, no nível de abstração e no nível de implementação. Este processo possibilita uma tradução com eficiência de um algarítmo para a linguagem da máquina.

Em um circuito digital observamos muitas diferenças de um modelo sequencial de processamento, pois em um sistema digital típico as partes são menores, e suas entradas e saídas possuem conexão entre si. No momento em que um sinal muda, as partes ligadas a ele se tornam ativas e se inicia um conjunto de novas operações que ocorrem de modo paralelo, levando cada uma um tempo diferente representando assim o atraso da propagação para completar. E no momento que completadas as operações as saídas das partes se tornam atualizadas.

No caso de o valor da saída ser modificado, será ativado por ele uma nova rodada de operações. Com essa descrição podem ser observadas características típicas de circuitos digitais, com a inclusão de entradas e saídas, operações paralelas, atrasos de propagação entre outros. O modelo sequencial de programação não é capaz de representar tais características e por esse motivo existem as linguagens HDL.

As funções mais importantes de um código em HDL são:

- Documentação: Uma vez que a linguagem de representação humana muitas vezes possui mais de uma interpretação, uma descrição HDL pode ser usada formalmente para especificar um sistema digital.
- Entrada para um simulador: Para realizar testes em um circuito sem necessariamente construí-lo, utiliza-se a simulação. Um simulador HDL fornece uma ferramenta para modelar o comportamento de um circuito lógico que opera em paralelo, em um computador cujas operações são sequenciais. Além da ferramenta é necessário um conjunto de arquivos contendo uma descrição em HDL do circuito externo ao que será testado, assim como estímulos. Este conjunto de arquivos é

chamado de testbench. Durante execução, o simulador interpreta o código HDL e gera respostas.

- Entrada para uma ferramenta de síntese (synthesizer): um programa de síntese recebe um programa HDL como entrada e com base nas bibliotecas, cria o circuito.

A linguagem de descrição de hardware VHDL (Very High Speed Integrated Circuit Hardware Description Language) foi utilizada para a programação do FPGA contido no kit de desenvolvimento. Assim como o GPS, o VHDL foi desenvolvido pelo Departamento de Defesa dos Estados Unidos por volta dos anos 80. Desde seu desenvolvimento o VHDL foi concebido para se tornar um padrão, o que atualmente sendo regulamentado pela IEEE.

A descrição de um circuito lógico digital em VHDL pode ser feita em nível RTL ou de portas lógicas. Quando construído em nível de portas lógicas, o circuito precisa ser projetado em mais baixo nível de abstração, sendo que deve ser descrito por portas lógicas (AND, OR, NOT) e elementos de memória (flip-flops, latches). Já quando descrito em nível de registradores (Register Transfer Level), o projeto pode ser feito em mais alto nível, sendo que elementos como somadores e registradores são os blocos de construção. O comportamento do circuito digital é descrito por máquinas de estado, e os dados são agrupados e interpretados por tipos de mais alto nível como estado ou inteiro.

A seguir temos um exemplo de descrição de um circuito digital de detecção de paridade par em VHDL. Os componentes descritos no capítulo 5 foram traduzidos para o VHDL seguindo estrutura semelhante.

```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity even_detector is

  port (
    a    : in  std_logic_vector;
    even : out std_logic);

end entity even_detector;

architecture sop_arch of even_detector is
  signal p1, p2, p3, p4 : std_logic;
begin  -- architecture sop_arch
  even <= (p1 or p2) or (p3 or p4);
  p1   <= (not a(0)) and (not a(1)) and (not a(2));
  p2   <= (not a(0)) and a(1) and a(2);
  p3   <= a(0) and (not a(1)) and a(2);
  p4   <= a(0) and a(1) and (not a(2));
end architecture sop_arch;
```

Figura 17 Circuito de detecção de paridade par em VHDL

Como se pode observar na Figura 17, o código consiste de duas grandes unidades: *entity* e *architecture*. O “entity” especifica as portas de entrada e saída do circuito. A unidade “architecture” especifica a operação ou organização interna do circuito.

Utilizar uma linguagem de descrição de hardware como VHDL é similar a utilizar um circuito real. Em ambos os casos conectamos as entradas ao circuito e observamos as suas saídas. A grande diferença é que ao utilizar o VHDL um circuito pode ser corrigido com a mesma facilidade de se escrever um texto em um editor de textos. Além disso, ao usar de simulações o projetista consegue analisar o funcionamento puro da lógica desenvolvida, evitando problemas oriundos de protótipos reais (curtos-circuitos, mal contatos, entre outros). Na **Figura 18** temos o “testbench” do detector de paridade par. Esta descrição faz o papel dos circuitos e componentes externos conectados ao projeto em desenvolvimento. Uma vez que esta descrição não será sintetizada, sendo somente utilizada em simulações, algumas diretivas (por exemplo “wait”) não sintetizáveis podem ser usadas.

Para verificação do funcionamento correto do módulo de comunicação I²C foram utilizados testbenchs escritos em VHDL que simulavam o funcionamento do oscilador Si599 como escravo na comunicação. Este funcionamento simulado foi baseado no comportamento esperado conhecido através do datasheet do oscilador.

```
library ieee;
use ieee.std_logic_1164.all;
entity even_detector_testench is
end even_detector_testbench;

architecture tb_arch of even_detector_testbench is
  component even_detector
    port(
      a: in std_logic_vector(2 downto 0);
      even: out std_logic;
      end component;
      signal test_in: std_logic_vector(2 downto 0);
      signal test_out: std_logic;
      begin
        -- instantiate the circuit under test
        uut: even_detector
          port map(a=>test_in, even=>test_out);
        -- test vector generator
        process
        begin
          test_in <= "000";
          wait for 200 ns;
          test_in <= "001";
          wait for 200 ns;
          test_in <= "010";
          wait for 200 ns;
          test_in <= "011";
          wait for 200 ns;
        end process;
        -- verifier
        process
          variable error_status: boolean;
        begin
          wait on test_in;
          wait for 100 ns;
          if((test_in="000" and test_out='1') or
             (test_in="001" and test_out='0') or
             (test_in="010" and test_out='0') or
             (test_in="011" and test_out='1'))
          then
            error_status := false;
          else
            error_status := true;
          end if;
          -- error reporting
          assert not error_status
            report "test failed."
            severity note;
        end tb_arch;
```

Figura 18 Testbench do circuito detector de paridade par em VHDL

É importante garantir que o sintetizador ira conseguir interpretar o circuito projetado para ser utilizado no FPGA. Alguns circuitos e declarações de componentes presentes no kit de desenvolvimento, mas fora do FPGA, possuem maneiras especificas de serem descritos. Na **Figura 19** está demonstrado um modelo de declaração de registradores. Os registradores utilizados pelo clock

disciplinado foram descritos em VHDL utilizando estrutura semelhante à mostrada na figura.

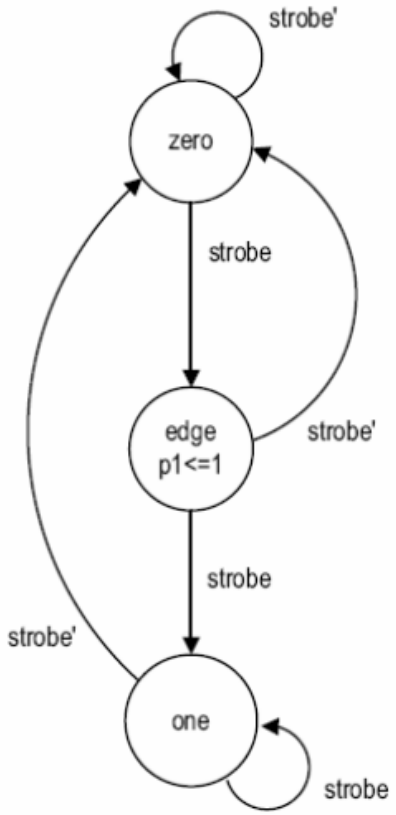
```
-- registers
process(clk, reset)
begin
  if reset = '1' then
    state_reg         <= idle;
    key_reg           <= (others => '0');
    val_reg           <= (others => '0');
    bytes_counter_reg <= (others => '0');
  elsif rising_edge(clk) then
    state_reg         <= state_next;
    key_reg           <= key_next;
    val_reg           <= val_next;
    bytes_counter_reg <= bytes_counter_next;
  end if;
end process;
```

Figura 19 Declaração de registradores em VHDL

Na Figura 20está o código que representa uma máquina de estados síncrona.

O código é dividido em três partes como no modelo de máquinas de estado síncronas: declaração de registradores de estado, lógica de próximo estado e lógica de saída. As máquinas de estado síncronas que realizam os circuitos descritos nas máquinas de estados do capítulo 5 foram descritas em VHDL seguindo estrutura semelhante à mostrada na Figura 20, porém adicionando os sinais, registradores, estados e outras particularidades específicas de cada circuito.

Existem muitos outros detalhes de semântica e sintaxe em VHDL, o qual este documento não entrará em detalhe. Mais sobre a linguagem pode ser encontrado em [11].




```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity edge_detector1 is
  port (
    clk, reset : in  std_logic;
    strobe     : in  std_logic;
    p1         : out std_logic);
end entity edge_detector1;

architecture moore_arch of edge_detector1 is
  type state_type is (zero, edge, one);
  signal state_reg, state_next : state_type;
begin  -- architecture moore_arch
  -- state register
  process (clk, reset) is
  begin  -- process
    if reset = '1' then
      state_reg <= zero;
    elsif clk'event and clk = '1' then
      state_reg <= state_next;
    end if;
  end process;
  -- next state logic
  process (state_reg, strobe) is
  begin  -- process
    case state_reg is
      when zero =>
        if strobe = '1' then
          state_next <= edge;
        else
          state_next <= zero;
        end if;
      when edge =>
        if strobe = '1' then
          state_next <= one;
        else
          state_next <= zero;
        end if;
      when one =>
        if strobe = '1' then
          state_next <= one;
        else
          state_next <= zero;
        end if;
    end case;
  end process;
  -- moore output logic
  p1 <= '1' when state_reg = edge else '0';
end architecture moore_arch;
```

Figura 20 Exemplo de máquina de estados em VHDL

6.2.1: Programando um FPGA

O processo de programação de um FPGA pode ser visualizado na **Figura 21** temos o processo de programação de um FPGA. Este processo foi utilizado para programar, a partir do código VHDL criado, o FPGA do kit de desenvolvimento para que realizasse a arquitetura mostrada na seção 5.1.

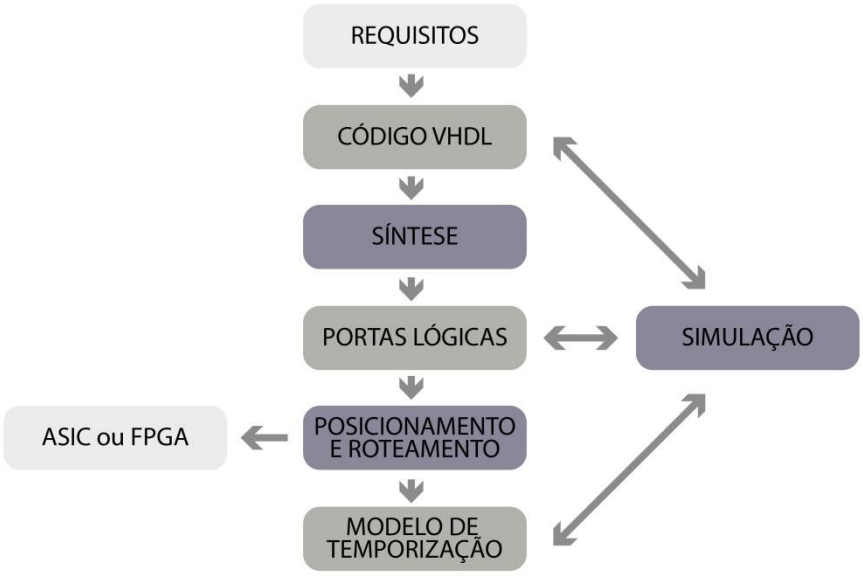


Figura 21 Processo de Programação de um FPGA

O processo começa com entrada de uma descrição em VHDL contendo as funcionalidades do projeto. Essa descrição é feita em RTL.

Esse código em VHDL passa por uma síntese, gerando uma “netlist”. Essa netlist contém um código VHDL agora em nível de porta lógica (Gate Level).

O processo segue com o roteamento e posicionamento dos blocos lógicos. Nesta etapa o layout do circuito combinacional é estabelecido, assim como dados de sua temporização.

Por fim, o FPGA é efetivamente programado para gerar o circuito estabelecido anteriormente. Nesta etapa os blocos lógicos são configurados e interconectados, formando o circuito projetado.

6.2.2: Programando o oscilador Si599

A frequência de saída (f_{out}) é determinada pela programação da frequência do DCO (f_{dco}) e os divisores de saída (HS_DIV e N1). A frequência de saída é calculada usando a seguinte equação:

$$\text{fout} = \frac{fdco}{\text{divisores de saída}} = \frac{fxtal * RFREQ}{HSDIV * N1}$$

A frequência do DCO é ajustada no intervalo de 4,85 à 5,67 GHz pela alteração do multiplicador fracionário de alta resolução (RFREQ). A frequência do DCO é o produto da frequência fixa do cristal interno (f_{xtal}) e RFREQ.

Os 38 bits de precisão do RFREQ permitem que o DCO tenha uma resolução de frequência programável de 28 ppt.

Como mostrado na Figura 22, o oscilador permite a reprogramação da frequência do DCO em até ±3500 ppm da frequência central configurada sem a interrupção do clock de saída. Mudanças maiores que esta janela de ±3500 ppm causarão uma mudança nas configurações internas e forçarão que o clock de saída pare momentaneamente e volte em qualquer momento arbitrário durante um ciclo de clock.

Este processo de recalibração estabelece uma nova frequência central e pode levar até 10ms. Circuitos sensíveis a variações ou pulsos devem ser resetados após o processo de recalibração ter sido concluído.

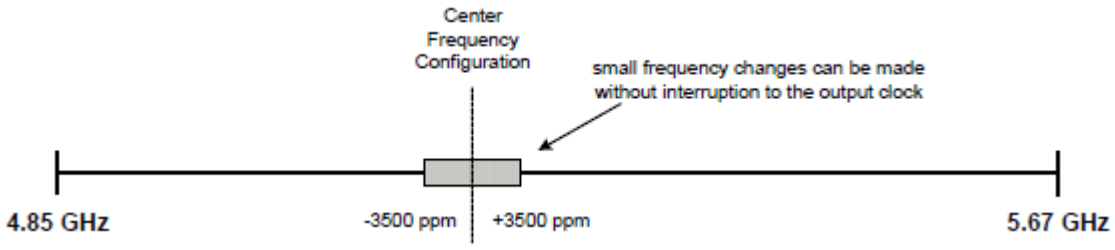


Figura 22 Janela para ajuste fino

6.2.3: Reconfigurando o clock de saída para pequenas variações na frequência

Para variações de frequência de saída inferiores a ±3500 ppm da frequência central, A frequência do DCO é o único valor que precisa ser reprogramado. Uma vez que $f_{dco} = f_{xtal} * RFREQ$ e f_{xtal} é fixo, mudar a frequência do DCO é a mesma coisa que mudar o valor de RFREQ como descrito a seguir:

1. Usando a porta serial I²C, ler o valor atual de RFREQ (registradores 0x08-0x12)
2. Calcular o novo valor de RFREQ utilizando a seguinte equação:

$$RFREQnew = RFREQcurrent * \frac{Fout_{new}}{Fout_{curent}}$$

3. Usando a porta serial I²C, escrever o novo valor de RFREQ (registradores 0x08-0x12). Mudanças multi-byte no RFREQ podem congelar o DCO para evitar valores de RFREQ indesejados.

Exemplo:

Um Si599 gerando uma frequência de saída de 148,35MHz precisa ser configurado em tempo real para gerar um clock de saía de 148,5MHz. Isso representa uma variação de +1011,122ppm, o que está dentro da janela de ±3500 ppm.

Uma típica configuração de frequência para esse exemplo:

$RFREQ_{current}$ = 0x8858199E9

$F_{out\_current}$ = 148,35MHz

F_{out_new} = 148,5MHz

Calcular $RFREQ_{new}$ para mudar a frequência de saída de 148,35 para 148,5MHz:

$$RFREQnew = 0x8858199E9 * \frac{148{,}5MHz}{148{,}35MHz}$$

$$= 0x887B6473C$$

É importante notar que para realizar o calculo do novo RFREQ é necessário no mínimo 38 bits de precisão.

Mudanças relativamente pequenas na frequência de saída podem exigir a escrita e mais de um registrador. Essas escritas multi-registrador podem impactar na frequência de saída.

Esses ruídos na frequência de saída podem ser prevenidos adotando-se os seguintes procedimentos:

1. Congelar o valor “M” (setar o bit 5 do registrador 135 = 1)
2. Escrever a nova configuração de frequência (RFREQ)
3. Descongelar o valor “M” (setar o bit 5 do registador 135 = 0)

6.2.4: Reconfigurando o clock de saída para grandes variações na frequência

Para variações de frequência fora da janela de ±3500 ppm da frequência central é necessário que tanto a frequência do DCO quanto os divisores de saída sejam reprogramados. Mais uma vez, é importante notar que mudanças maiores que esta janela de ±3500 ppm causarão uma mudança nas configurações internas e forçarão que o clock de saída pare momentaneamente e volte em qualquer momento arbitrário durante um ciclo de clock.

O processo de reconfiguração da frequência de saída para variações fora da janela de ±3500 ppm é descrito à seguir:

1. Usando a porta serial I²C, ler os valores atuais de FREQ, HSDIV e N1.
2. Calcular f_{xtal} do oscilador. Importante notar que devido a pequenas variações na frequência do cristal de um oscilador para outro, cada oscilador pode ter um valor diferente de RFREQ ou até mesmo valores

diferentes de HSDIV e N1 para manter a mesma frequência de saída. Sendo assim, é necessário calcular o f_{xtal} de cada oscilador.

$$Fxtal = \frac{fout * HSDIV * N1}{RFREQ}$$

Uma vez que o valor de f_{xtal} tenha sido determinado, os valores de RFREQ, HSDIV e N1 são calculados para gerar a nova frequência de saída (f_{out}).

O primeiro passo para se calcular a configuração de frequência manualmente é determinar os novos valões dos divisores (HSDIV e N1). Dada a frequência de saída desejada, deve-se encontrar valores para os divisores de frequência que mantenham a frequência de DCO entre o intervalo de 4,85 e 5,67 GHz.

$$fdco\_new = fout\_new * HSDIVnew * N1new$$

Valores válidos para HSDIV são 9 e 11. O valor de N1 pode ser 1 ou qualquer número par até 128 (1,2,4,6,8,...,128). Para ajudar a minimizar o consumo de energia do oscilador, o divisor deve ser selecionado para manter a frequência do DCO o mais baixa possível. O menor valor de N1 e o maior valor de HSDIV resultam no menor consumo de energia.

Uma vez que HSDIV e N1 tenham sido determinados, o próximo passo é calcular o multiplicador da frequência de referência (RFREQ).

$$RFREQnew = \frac{fdco_{new}}{fxtal}$$

RFREQ é programado como um número fracionário de 38 bits onde os 10 bits mais significativos representam a parte inteira do multiplicador e os últimos 28 bits representam a parte fracionária.

Antes de entrar um número fracionário no registrador RFREQ, ele precisa ser convertido para um número inteiro de 38 bits usando a operação de deslocamento para a esquerda (shift left) de 28 bits, o que efetivamente multiplica RFREQ por 2^{28}.

Exemplo 1 (detalhando os procedimentos de ajuste):

RFREQ = 136,3441409d

RFREQ multiplicado por 2^{28} = 36599601635,42d

Descartando a parte fracionária = 36599601635d

Convertendo para hexadecimal = 0x8858199E9

Uma vez que os valores de RFREQ, HSDIV e N1 estejam determinados, eles podem ser escritos diretamente através da porta serial I²C de acordo com o seguinte procedimento:

1. Congelar o DCO (bit 4 do registrador 137)
2. Escrever a nova configuração de frequência (registradores RFREQ, HSDIV e N1)
3. Descongelar o DCO (bit 4 do registrador 137)
4. Setar o NewFreq bit (bit do registrador 135).

Exemplo 2 (detalhando os cálculos necessários):

Um Si599 gerando uma frequência de saída de 156,25MHz precisa ser reconfigurado para gerar uma frequência de saída de 161,1328125MHz. Essa alteração é maior que a janela de ±3500 ppm.

Fout = 156,25MHz

Leitura dos valores atuais de RFREQ, HSDIV e N1:

RFREQcurrent = 0x7FA611E85 = 34265439877d

34265439877d / 2^{28} = 127,64871074631810d

HSDIV = 4

N1 = 8

Calcular f_{xtal} e $f_{dco\_current}$:

$$Fdco\_current = fout * HSDIV * N1 = 5{,}000000000GHz$$

$$Fxtal = \frac{fdco_{current}}{RFREQcurrent} = 39{,}17MHz$$

Para fout_new = 161,1328125 MHz, escolher os divisores de saída que manterão a frequência do DCO (fDCO) entre o intervalo de 4,85 à 5,67 GHz. Neste caso particular, manter os mesmos valores dos divisores de saída já mantém a frequência do DCO no intervalo desejado:

$$Fdco\_new = fout\_new * HSDIVnew * N1new$$

$$= 161{,}1328125\ MHz * 4 * 8 = 5{,}156250000\ GHz$$

Calcular o novo valor de RFREQ utilizando o novo valor de fdco:

$$RFREQnew = \frac{fdco_{new}}{fxtal} = 131{,}67733d = 0x83A342779$$

6.2.5: Comunicação por porta I²C

A interface de controle do Si599 é compatível com o barramento I²C de 2 fios para comunicação bidirecional. Este barramento consiste em uma linha serial de dados (DAS) e um clock serial de entrada (SCL). Ambos fios devem ser conectados a uma tensão positiva através de resistores de pullup. O modo de operação rápido é suportado para taxas de transferência de até 400kbps como especificado nos padrões de comunicação I²C.

A Figura 1 exemplifica os comandos para acesso de escrita e leitura. O tamanho do dado é sempre de 1 byte. Tanto escrita quanto leitura suportam

transmissão de 1 ou mais bytes como ilustrado. O mestre precisa enviar um “not ack” e um sinal de “stop” após o último dado lido para terminar o comando de leitura.

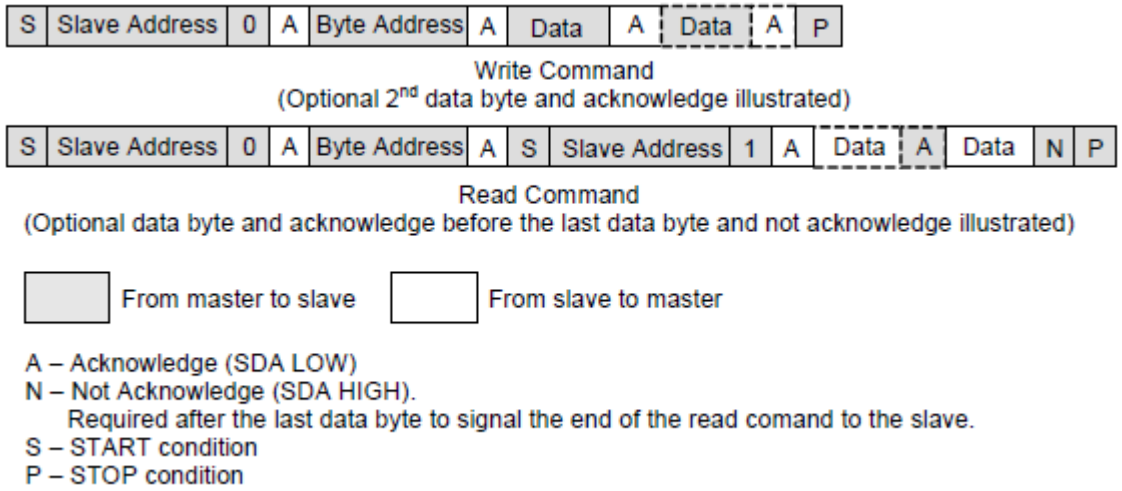


Figura 23 Leitura e escrita I²C

6.3: Ferramentas utilizadas

Para compilação do código em VHDL e programação do FPGA foi utilizado o programa Quartus II da Altera (Figura 24). O Quartus também realiza análise temporal para verificar se os requisitos temporais dos elementos de memória são respeitados e se os atrasos de propagação estão dentro do aceitável para uma determina frequência.

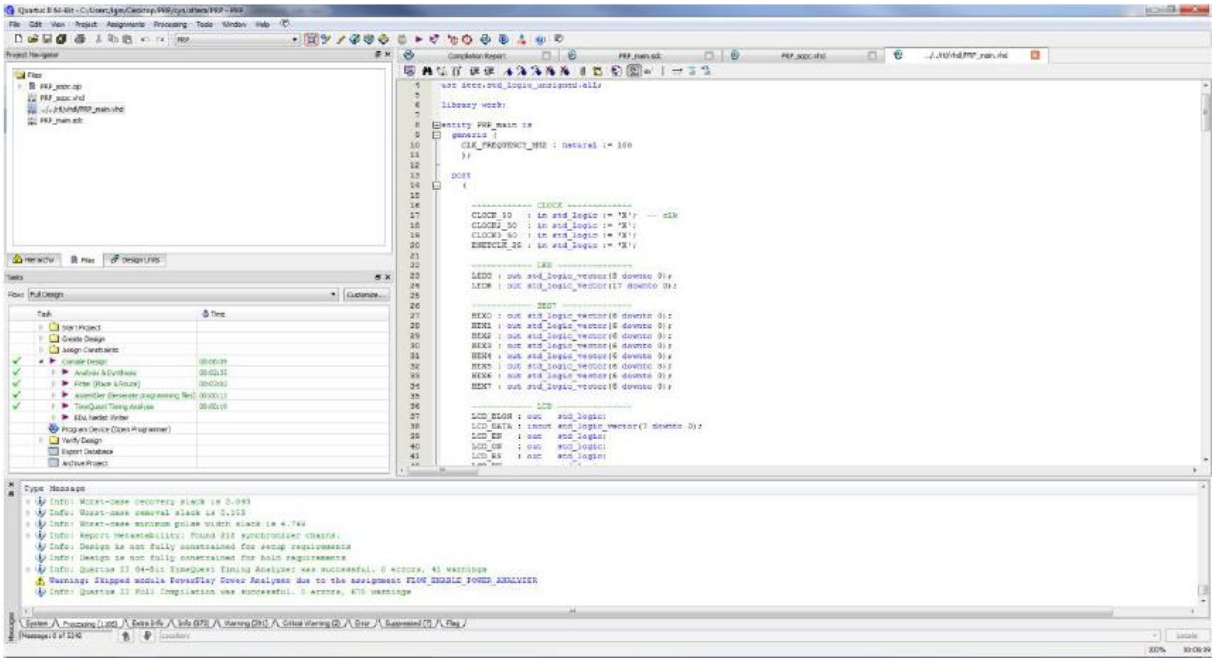


Figura 24 Ambiente Quartus II

Para simulação do funcionamento dos circuitos digitais foi utilizado o software ModelSim da Mentor Graphics. Esse software realiza suas simulações baseado em testbenchs escritos em VHDL, e possui uma interface gráfica (Figura 25) para mostrar os sinais gerados na simulação.

Para testes de verificação dos dados sendo transmitidos através da comunicação I²C, foi utilizada a ferramenta SignalTap (Figura 26). O SignalTap é uma ferramenta de análise dos registradores no projeto desenvolvido no FPGA, que realiza a captura os valores de registradores específicos a cada vez que uma condição determinada seja atingida. Ele foi utilizado nesse trabalho para verificar se o Si599 estava se comunicando corretamente com o FPGA e principalmente para verificar se as leituras e cálculos estavam corretos.

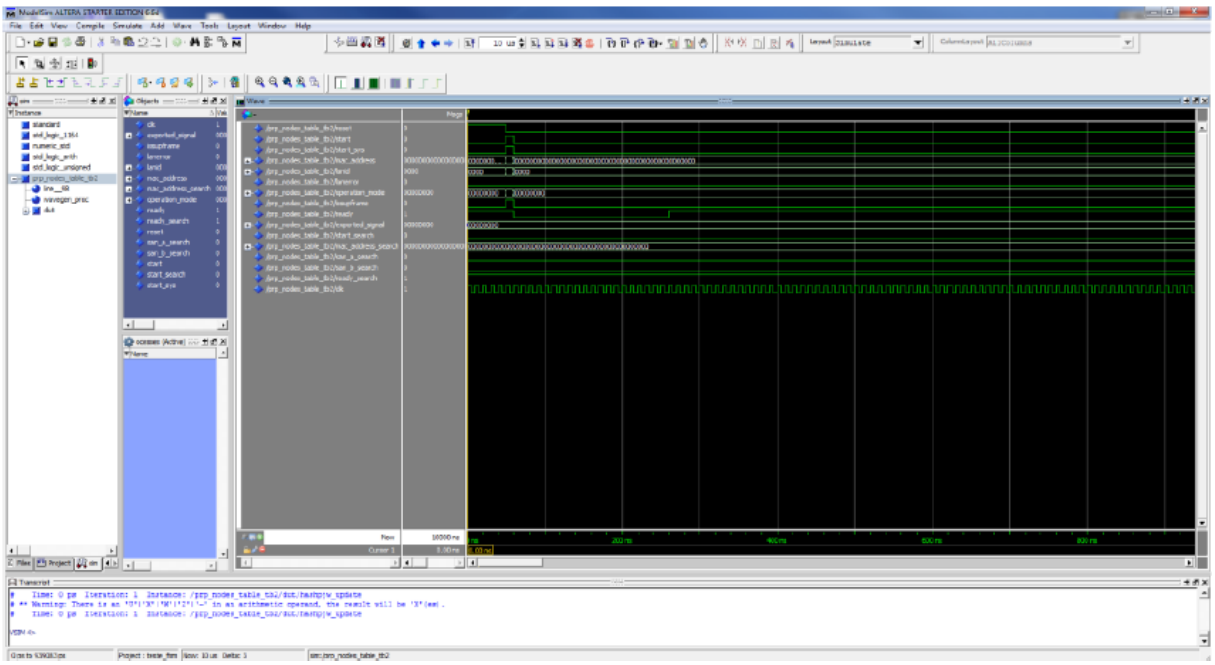


Figura 25 Ambiente ModelSim

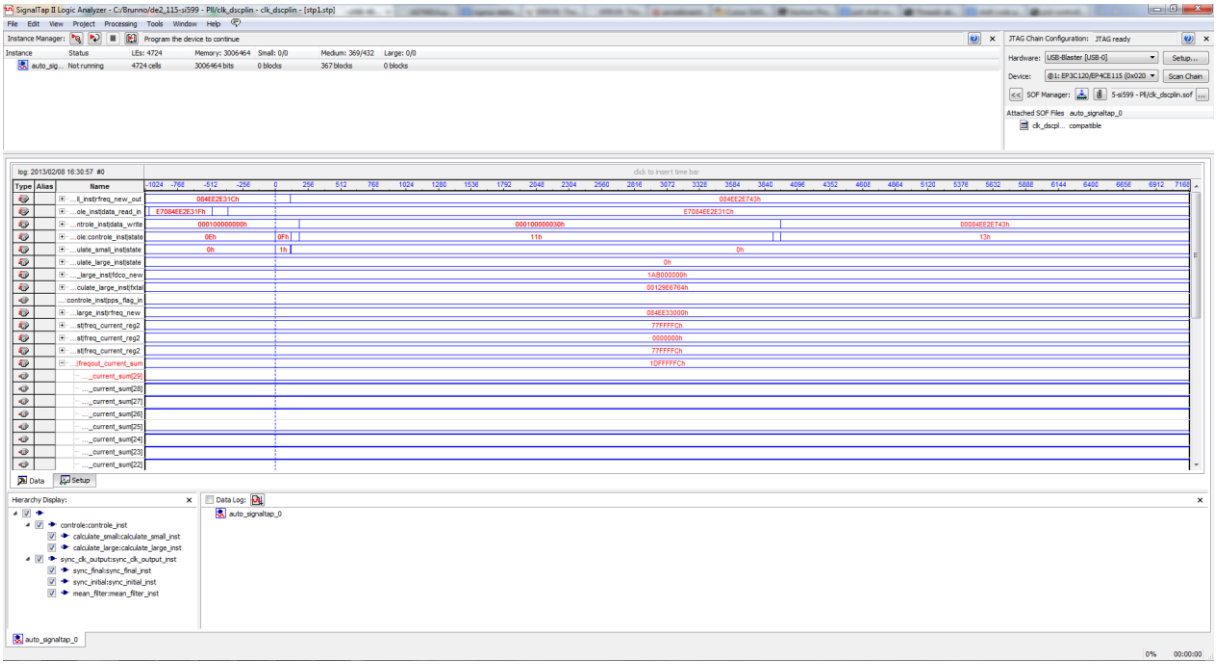


Figura 26 Ferramenta SignalTap

6.4: Testes de comunicação

O FPGA do kit de desenvolvimento foi programado para realizar a arquitetura descrita na seção 5.1. Contudo, para garantir que o módulo I²C desenvolvido estivesse funcionando corretamente era necessário que se tivesse o controle de todo o comportamento tanto do mestre, quanto do escravo.

Para realizar esse teste foi criado um testbench capaz de simular o comportamento esperado do oscilador Si599 em uma comunicação através do barramento I²C. Esse comportamento desejado foi obtido do estudo do datasheet do oscilador como exemplificado na seção 6.2.5:.

O módulo I²C foi programado inicialmente apenas para escrever dados variando a quantidade de bytes enviados. O Testbench neste primeiro momento apenas recebia os dados e enviava “acks” conforme fosse necessário. Com a simulação no ModelSim os problemas de mal posicionamento das bordas de subidas e consequentemente falhas de comunicação puderam ser facilmente observados e corrigidos.

Em um segundo momento foi implementado a leitura de dados através do barramento I²C. Nesta etapa ouve um grande trabalho no testbench, uma vez que este não precisava apenas mandar sinais de “ack” a cada 8 bits, mas também precisava interpretar os dados a fim de saber qual procedimento estava sendo realizado(escrita ou leitura).

Este empenho para garantir uma comunicação correta levou não apenas ao desenvolvimento do mestre projetado para ser parte do projeto, mas também no desenvolvimento de um escravo do protocolo de comunicação I²C totalmente funcional e ainda mais complexo de ser projetado. O projeto do escravo é ligeiramente mais complexo, pois ao contrário do mestre, o escravo não sabe o que

ira acontecer e precisa responder ao mestre no ciclo de clock seguinte a requisição, como descreve o protocolo.

Com a garantia de funcionamento da comunicação I²C por simulação, o FPGA foi reconfigurado para realizar a comunicação com o oscilador Si599 de fato. Esta foi possivelmente a parte mais demorada do desenvolvimento de todo o projeto. Devido a um curto-circuito na placa do Si599, o cabo do barramento de dados do barramento I²C estava aterrado. Como o primeiro teste realizado foi o teste de escrita, por se tratar de um procedimento mais simples, ao final de cada byte escrito o mestre esperava um sinal de “ack” e como o barramento estava aterrado, ele reconhecia esse erro como sendo um sinal lógico zero e consequentemente um “ack”. Com isso continuava as escritas normalmente sempre recebendo “acks” falsos.

Já durante a leitura de dados o problema era inverso e começou a ser possível a sua identificação. Como no procedimento de leitura é necessário realizar escritas inicialmente, a comunicação se dava “sem problemas” até o momento onde a leitura era efetivamente realizada. Neste momento todos os dados obtidos eram o valor lógico zero, porém como em teoria esse é um dado válido, a leitura prosseguia até sua conclusão e os LEDs sinalizadores de bom funcionamento eram acesos.

Este problema só foi identificado e isolado depois de utilizado o módulo I²C do kit de desenvolvimento. O mestre projetado realizou uma comunicação com o componente I²C do kit de desenvolvimento, onde a leitura e a escrita foram concluídas com êxito.

Com a certeza de que o projeto em VHDL estava correto e com a ajuda do osciloscópio o curto-circuito foi identificado e a comunicação com o oscilador Si599 foi estabelecida.

Um novo problema surgiu na comunicação durante os próximos testes porém este problema será abordado aqui.

Durante a leitura em alguns momentos todos os dados lidos eram o nível lógico 1. Uma nova investigação foi realizada para eliminar esse problema, já que a consistência dos dados é fundamental para se calcular a atuação necessária do controle. Depois de intensa pesquisa foi descoberto que os resistores de pullup

usados no projeto da placa com o oscilador possuíam resistência menor que a especificada no protocolo de comunicação I²C. Isto fazia com que o barramento que é ligado através desses resistores a uma fonte de tensão positiva estivesse em algumas ocasiões em alta impedância. Com a substituição dos resistores o problema foi contornado, porém ainda ocorre raramente.

Para ter uma completa confiança nos dados foi introduzido uma condição no módulo de leitura que descarta dados inválidos (terminados em 0x”FFF”).

6.5: Testes de ajuste automático

Com a comunicação funcionando, foi possível passar para a etapa de testes das estratégias de controle.

A primeira tentativa de controle foi utilizar um algoritmo de controle PID desenvolvido em VHDL. Essa abordagem é por muitos motivos a melhor abordagem de controle em teoria neste caso, entretanto o oscilador apresentava comportamentos inesperados.

E na tentativa de simplificar o problema, a estratégia de controle descrita na seção 6.2.2:.

Utilizando o SignalTap exaustivamente não foi possível identificar os motivos que levavam o oscilador a não se comportar como o esperado. Durante a escrita do registrador RFREQ nenhum problema acontecia, já na leitura do mesmo registrador no ciclo seguinte, a informação contida no registrador não era a mesma que acabará de ser gravada. Os valores lidos nos registradores eram sempre os mesmos não importando o que fossem escritos neles.

Com prazo se esgotando e os testes de ajuste automático não surtindo efeito uma nova estratégia foi necessária.

6.6: Testes de ajuste manual

Durante os testes de ajuste automático, por um e apenas um ciclo de um segundo entre dois sinais pps, a frequência de saída se aproximou da desejada e a leitura dos registradores mudou para outro valor diferente do que sempre havia ocorrido até então.

Com esta nova informação sobre o comportamento do oscilador surgiu a ideia de setar valores para registradores ao invés de usar as metodologias já descritas.

Isto tornou possível identificar que o oscilador não é capaz de gravar qualquer valor para RFREQ para qualquer combinação de divisores.

Com ajustes empíricos e cálculos feitos manualmente baseados nos procedimentos descritos no datasheet do oscilador foi possível achar um ajuste inicial para os registradores que deixa a frequência de saída próxima da desejada.

Uma vez que essa frequência esteja próxima da desejada a realização do ajuste automático para grandes variações de frequência, garante que este ajuste inicial manual não seja mais usado e assim a variação da frequência do cristal de cada oscilador pode ser compensada normalmente.

Com este ajuste o projeto teve o seu funcionamento correto e dentro dos requisitos desejados.

6.7: Recursos utilizados

Após a compilação do código VHDL, o software Quartus II disponibilizou um relatório de recursos utilizados do FPGA pelo circuito lógico do clock disciplinado. Os principais dados foram resumidos na Tabela 1.

Tabela 1 Recursos Utilizados

Descrição	Utilizado	Total	Percentual
Elementos lógicos totais	3.334	114.480	3%
Total de elementos lógicos combinacionais	2.702	114.480	2%
Registradores lógicos dedicados	2.252	114.480	2%
Total de registradores	2.252	-	-
Total de pinos	17	529	3%
Total de bits de memória	168	3.981.312	<1%
Multiplicadores de 9 bits embarcados	53	532	10%
Total de PLLs	1	4	25%

6.8: Resultados

Como resultado final do projeto descrito no presente trabalho, foi obtido um circuito gerador de clock (clock disciplinado) que atende as especificações exigidas para o projeto.

Atualmente o erro máximo a cada sinal de pps é de 72ns o que atende o padrão IEEE1588, que exige que o erro seja inferior a 100ns.

O clock disciplinado também é capaz de compensar as variações do cristal, como por exemplo, a variação de temperatura. Garantindo sempre a precisão mencionada.

Com a volta do sinal de referência o clock disciplinado consegue voltar a seguir a referência instantaneamente ou limitando o salto máximo, que pode ser configurado dependendo da necessidade.

6.9: Considerações

Devido a novas descobertas e perspectivas de melhoria na precisão do projeto do clock disciplinado, a proteção que garante o funcionamento sem saltos com a volta do pps foi removida temporariamente do projeto. Isto se deu para facilitar a depuração de erros em novos testes realizados com novas abordagens do problema. Como exemplo de abordagens diferentes posso citar a inclusão de um filtro "média" no pulso pps para diminuir as oscilações de recepção de pulsos pps.

Um outro exemplo seria a utilização do próprio clock disciplinado como clock do sistema após este clock estar em regime permanente. Essa abordagem garantiria uma total independência ao clock do oscilador do kit de desenvolvimento em regime permanente.

Para reabilitar a função de proteção que evita os saltos de frequência, basta incluir um saturador de limite σ (salto máximo permitido) no ajuste do registrador RFREQ.

Capítulo 7: Conclusões e Perspectivas

Nesta monografia foi proposta e implementada uma arquitetura de circuito digital em FPGA de um clock disciplinado com seguimento de referência. O FPGA foi programado utilizando a linguagem VHDL e ao longo do documento, foi apresentada a arquitetura do circuito digital do clock disciplinado e a descrição do seu funcionamento. O ajuste automático realizado no clock disciplinado foi o proposto no datasheet do oscilador utilizado e a precisão da medição da frequência de saída foi aumentada utilizando um pll.

Testes realizados com o auxílio das ferramentas ModelSim e SignalTap, comprovaram que o circuito proposto e criado nessa monografia para o clock disciplinado, atende os requisitos estabelecidos para funcionamento desejado: se ajustar automaticamente e continuamente em tempo real para a obtenção de uma frequência fixa de saída sem “saltos” a cada ajuste.

Apesar do módulo de clock disciplinado criado funcionar corretamente, ele não está pronto para a inclusão nos produtos desenvolvidos para empresa. Ainda é necessária melhoria da comunicação I²C entre o oscilador e o FPGA, assim como uma nova placa deve ser projetada para suportar uma comunicação através do barramento HSMC.

Com a inclusão do clock disciplinado nos produtos da empresa, espera-se que esses produtos ganhem mais um diferencial que os tornem mais atraentes para o mercado, conquistando novos clientes e ajudando no crescimento da empresa. Além disso a melhoria na precisão abre portas para que a empresa possa ingressar no setor de telecomunicações.

A realização desse trabalho foi de fundamental importância para formação do estudante como engenheiro de controle e automação, ao permiti-lo ter uma experiência de trabalho em ambiente profissional empresarial, possibilitando a interação com problemas reais de uma empresa e a busca por soluções.

Bibliografia:

[1] ONS. Operador Nacional do Sistema. Acesso em 10 de Fevereiro de 2013, disponível em: http://www.ons.org.br/institucional/modelo_setorial.aspx

[2] IEC. IEC 61000-4-30: Testing and Measurement Techniques – Power Quality. McGraw-hill, 1996.

[3] IEEE1588 (Julho de 2008) Standard for Precision Clock Synchronization Protocol for Networked Measurement and Control Systems, New York

[4] Reason Tecnologia S/A. (s.d.). A Empresa. Acesso em 07 de Fevereiro de 2013, disponível em Reason Tecnologia S/A: http://www.reason.com.br/pt/a-reason/a-empresa.

[5] Reason Tecnologia S/A. (s.d.). Produtos. Acesso em 16 de Fevereiro de 2013, disponível em Reason Tecnologia S/A: http://www.reason.com.br/pt/produtos.

[6] FRAGA, J. Sincronização e Ordenação – Sistemas Distribuídos DAS5315 – Notas de Aula, 2012

[7] Si598/Si599 10-810 MHz Programmable XO/VCXO – Datasheet. Disponível em http://www.silabs.com/Support%20Documents/TechnicalDocs/si598-99.pdf

[8] I²C-Bus Specification and User Manual, Rev. 5, October 2012. Disponível em http://www.nxp.com/documents/user_manual/UM10204.pdf

[9] WAKERLY, J. F. (s.d.). Digital Design: Principles & Practices (3ª ed.). Prentice Hall.

[10] Altera Corporation. (Maio de 2011). User Guide. San Jose, CA.

[11]CHU, P. P. (2006). RTL Hardware Design Using VHDL: Coding for Efficiency, Portability and Scalability. New Jersey: Wiley-Interscience.